ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

SCHOOL OF INFORMATION SCIENCE

Towards Improving the Performance of Spontaneous Amharic Speech Recognition

By

TEWODROS GIZAW TOHYE

October 2015

ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES
SCHOOL OF INFORMATION SCIENCE

Towards Improving the Performance of Spontaneous Amharic Speech Recognition

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

By

TEWODROS GIZAW TOHYE

October 2015

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

SCHOOL OF INFORMATION SCIENCE

Towards Improving the Performance of Spontaneous Amharic Speech Recognition

By

TEWODROS GIZAW TOHYE

Name and Signature of Members of the Examining Board

Name	Title	Signature	Date
	Chair Person,	_____	_____
Dr.Martha Yifiru	Advisor,	_____	_____
Dr.Million Meshesha	Examiner.	_____	_____
Dr.Solomon Teferra	Examiner,	_____	_____

ACKNOWLEDGMENT

First of all, I would like to thank God for supporting and being with me in all walks of my life.

I would like to thank my advisor Dr. Martha Yifiru for her constructive comments, guidance leadership and support. I am thankful forher because without her guidance and genuine comments the completion of this research would have not been possible.

My special thanks go to Adugna and Bantegize for their spontaneous speech corpus and supports which helps me for this study.

I would like to thank Dr.SolomonTeferra since he provides me a text corpus for the language model, and Dr.SebsibeHailemariam for syllabification parser. W/t Adey Edessa you supported me a lot while I got difficulties in my work, thank you.

I am also grateful to my friends Hailemariam and IES staff members especially, Genet, Almaz, and Mahlet and others for their cooperation and willingness to use their office for internet connection, providing paper and unlimited material and moral support.

ABBREVIATIONS ANDACRONYMS

ASR	Automatic Speech Recognition
BR	Breath
CV	Consonant Vowel
FP	Filled Pause
HES	Hesitation
HMM	Hidden Markov Model
HTK	Hidden Markov Toolkit
INT	Interruption
LGH	Laugh
LM	Language Model
MFCC	Mel-Frequency Cepstrum Coefficients
NLP	Natural language processing
REP	Repetition
SASR	Spontaneous Automatic Speech Recognition
WER	Word Error Rate

Table of Contents

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

The ultimate goal of automatic speech recognition is towards developing a model that converts speech utterance into a sequence of words. With the objective of transforming Amharic speech into its equivalent sequence of words, this study explored the possibility of improving the performance of Amharic spontaneous speech recognition system using hidden Markov model (HMM).

To this end, four experiments have been conducted in order to improve the performance of the recognizer. The first three experiments were conducted using the spontaneous speech corpus consisting of 2007 sentences uttered by 36 people from different sex and age groups. This training data consists of 9460 unique words and it is around 3 hours and 10 minutes speech. For testing, speech of 104 sentences uttered by 14 speakers, consisting of 820 unique words has been used. The experiments have been conducted using different parameter tuning, and using CV-syllables and cross-word tri-phone as recognition units.

The fourth experiment has been done by increasing the corpus size. A speech corpus consisting of 3556 sentences uttered by 60 speakers from different sex and age group has been used for training. This training data consists of 12306 unique words and it is around 4 hours and 30 minutes of speech performance improvement has been achieved when cross-word tri-phone acoustic and tri-gram language models has been used in recognition. In this system, 58.67% words are correctly recognized, and 49.13% accuracy for the mixed test set, and 46.08% words are correctly recognized, and 32.42% accuracy for the speaker independent test set.

From the experimental result we found that using the tuning techniques, changing the sub-word unit using cross-word tri-phone, and tri-gram language model increases the performance of the recognizer. Even if the study come up with performance improvement there is a need to control the existing large variations in the realized speech waveform due to speaking variability, mood, and environment, in spontaneous speech rather than read speech. Besides these, the available spontaneous speech data set is small in size; so it is better to prepare large size spontaneous speech corpus by automatic transcription.

CHAPTER ONE
INTRODUCTION

1.1 Introduction

Speech is the most natural means of communication between humans it can be done without any tools or any explicit education. From previous several decades human beings tried to create technologies that could recognize correct speech [1]. While humans can differentiate speech very easily, they in fact make use of much acoustic, linguistic and contextual information. It has been seen that relation between physical speech signal and the corresponding words is so much complex and very hard to understand. Both the research areas of automatic speech recognition (ASR) and human speech recognition (HSR) observe the recognition process from the acoustic signal to a series of recognized units. For ASR, the objective is to automatically transcribe the speech signal in terms of a sequence of items as close as possible [1]. To an HSR, the attention is on understanding how human listeners recognize spoken utterances. On the basis of advances in statistical modeling of speech, automatic speech recognition (ASR) systems find extensive application in tasks that make use of human-machine interface, such as automatic call processing in telephone networks and query-based information systems that provide updated travel information, stock price quotations, weather reports, embedded systems etc.

Speech Recognition also known as Automatic speech recognition (ASR) is defined as a process of converting a speech signal into a set of words by a certain algorithm that can be implemented as a system program or a process of converting an acoustic signal, captured by a microphone or a telephone, to a set of words [2].

Automatic Speech Recognition as successive transformations of acoustic micro structure of speech signal into its implicit phonetic macro-structure. In other words, a speech recognition system is a speech-to-text conversion where in the output of the system displays text corresponding to the recognized speech [3]. Automatic Speech Recognition (ASR) is a technology that allows a computer to identify the words that a person speaks into a microphone or telephone [4]. As the most natural communication modality for humans, the ultimate dream of speech recognition is to enable people to communicate more naturally and effectively. Speech recognition is an interdisciplinary filed in nature, so doing research in this domain has its own challenges and opportunities.

As a challenge, since speech recognition system inexorably connected with the field of signal processing, physics(acoustic), linguistic and computer science, in order to build successful speech recognition system it requires knowledge and expertise from a wide range of disciplines. This expertise has a role in automatic speech recognition. The role of expertise in the field of signal processing is to extract relevant information from speech in an efficient and robust manner, physics to understand the acoustic environment well the human vocal tracts mechanisms and human hearing mechanism, linguists in order to understand the relationship between sound (phonology), word in language (syntax), meaning of spoken word (semantic) and sense derived from meaning (pragmatic) [1]. The other issuers which makes speech recognitions challenging task is phonological variations in both local and global context, individual differences, environmental factors (noise, transducers), diversity of language use and real-world issues (disfluencies, new words) [2]. Besides its challenges, the field of automatic speech recognition has witnessed a number of significant advances in the past 5 - 10 years, spurred on by advances in signal processing, algorithms, computational architectures, tools, techniques, approaches and hardware [5].These advances include the widespread adoption of a statistical pattern recognition paradigm, a data-driven approach which makes use of a rich set of speech utterances from a large population of speakers, the use of stochastic acoustic and language modeling, and the use of dynamic programming based search methods. Various approaches and types of speech recognition systems came into existence in last five decades gradually [5].

1.2 Classification of Automatic speech recognition

Speech recognition systems can be categorized based on different parameters; important of which are based on speaking Mode (isolated or discrete and continuous speech), based on enrollment (speaker-dependent and speaker-independent), based on Vocabulary size (small, medium and large), based on Language Model (context-sensitive and context in-sensitive) and based on speaking style (Dictation and Spontaneous speech).Based on speaking mode (utterance) speech is either discrete or continuous. Discrete speech consists of isolated words that are separated by silences [6]. The advantage of discrete speech is that word boundaries can be set exactly while with continuous speech; words will be spoken without silence [6].

The other category is based on the speaker enrollment it is either speaker dependent or independent. Speaker dependent systems are designed for a specific speaker [7]. They are generally more accurate for the particular speaker, but much less accurate for others speakers. These systems are usually easier to develop, cheaper and more accurate than speaker dependent, but not as flexible as speaker adaptive or speaker independent systems. Speaker independent models are designed for variety of speakers.

It recognizes the speech patterns of a large group of people. This system is most difficult to develop, most expensive and has less accuracy than speaker dependent Systems. However, they are more flexible. Based on the language model we want to apply speech is either context-sensitive or context insensitive. More general language models approximating natural language are specified in what is known as a context sensitive grammar.

Speech recognition systems that increase their accuracy by anticipating or limiting what can be said at any given time are referred to as context-sensitive recognition systems whereas systems that allow users to say anything, anytime without any constraint are called context insensitive recognition systems [29].Context-sensitive recognition systems are more difficult to implement on computers than context-insensitive ones. Based on speaking style automatic speech recognition is classified into either Read speech or Spontaneous speech. Spontaneous Speech is not well structured, acoustically and syntactically, as read speech [5]. Spontaneous Speech is a natural and basic form of day-to-day communication among people. Accurate recognition of this kind of speech is very desirable. However, its naturalness stirs up much more pronunciation variants in spontaneous speech than in read or prepared speech. There are also pause-fillers, laughter and other non-speech events, all of which make the recognition task difficult. Literature reveal that the accuracy rate for red speech is higher than that of spontaneous speech due to, the reason red speech and spontaneous speeches are different acoustically as well as linguistically. Spontaneous speech is characterized by having filled pauses, repairs, hesitations, repetitions, partial words, and dis-influences. There are three major groups of dis-influences in spontaneous speech that influence the quality and performance of any spontaneous speech recognition system [8]:

- Filled pauses (FP): short words, which appear as interjection – e.g.: uh, aaa. They are language dependent.

- Word repetitions: disfluencies used by the speaker to gain time before continuing with the sentence.

- Sentence restarts: speaker pronounces the initial part of a sentence and then starts over again with a new initial part.

1.3 Approaches in Automatic Speech Recognition

As noted by Anusuya and Katti [2], speech recognition process uses the following modeling (approaches).The first one is the Acoustic-phonetic approach. This approach is one of the earliest approaches to speech recognition and it is based on finding speech sounds and providing appropriate labels to these sounds [2]. It relies on explicit characteristics of speech sounds. In this approach the acoustic properties of phonetic unit is governed by straight forward rule to be understood by machine, but the acoustic properties of phonetic unit is not consistent since it is affected by different real world factors like speakers and neighboring phonetic unit [1]. In acoustic phonetic approach initially we have to do segmentation and labeling before proceeding to the next steps. Segmenting the speech signal into discrete (in time) region where the acoustic properties of the signal are representative of one (or possibly several phonetic unit or classes, and then attaching one or more phonetic labels to each segmented region according to the acoustic properties.

The second approach is Pattern Recognition approach, which is also known as pattern-matching approach, involves two essential steps namely, pattern training and pattern comparison [1]. The essential feature of this approach is that it uses a well formulated mathematical framework, robustness and invariance to different speech vocabularies in users, feature sets, tern comparison algorithms and decision rules. This property make it suitable for a wide range of speech unit and establishes consistent speech pattern representations, for reliable pattern comparison, from a set to labeled training samples via a formal training algorithm. The key idea of pattern recognition is to optimally extract patterns based on certain conditions and thereby separate the data into different classes. Moreover, pattern recognition approach performs classification without having any idea about the distribution of the measurements in different groups. There are two sub

approaches under pattern recognition approach the template based approach and stochastic approach.

The other is Template based approaches, which is categorized under the pattern recognition approach. In template based approach, unknown speech is compared against a set of per-recorded words (templates) in order to find the best match [4]. As a reference a set of template is stored and read and the comparison for the unknown utterance is made against the stored template. Recognition is carried out by selecting the category of the best matching pattern.

Further, Stochastic approach is also categorized under the pattern recognition approach. "Stochastic modeling entails the use of probabilistic models to deal with uncertain or incomplete information." [4]. In speech recognition, uncertainty and incompleteness arise from many sources; for example, confusable sounds, speaker variability's, contextual effects, and homophones words. Thus, stochastic models are particularly suitable approach to speech recognition. The most popular stochastic approach today is hidden Markov modeling. A hidden Markov model is characterized by a finite state Markov model and a set of output distributions. "It is a doubly stochastic process which has an underlying stochastic process that is not observable (hence the term hidden), but can be observed through another stochastic process that produces a sequence of observations." [2]. The Hidden Markov model (HMM) is a very powerful mathematical tool for modeling time series data. It provides efficient algorithms for state and parameter estimation, and it automatically performs dynamic time warping for signals that are locally squashed and stretched. It can be used for many purposes other than acoustic modeling [8]. This specific study also uses the stochastic approach and the well know speech recognition model Hidden Markov Model (HMM) is selected based on the above mentioned reasons. Finally the artificial intelligence approach is "a hybrid of the acoustic phonetic approach and pattern recognition approach." [1]. It took the advantage of the two approaches; the basic idea in this approach is to compile and integrate knowledge from a variety of knowledge source and to convey it on the problem at hand. The recognition procedure held in artificial intelligence approach is by mimic human's capability of visualizing, analyzing, reasoning and making decision to applying it to computer on the measured acoustic features.

1.4 Statement of the problem

In a day to day activity people communicate through speech. It is a focus area nowadays to make communication between people and machine through speech. Since the communication between people is using continuous (spontaneous) speech, the communication between human and computer must be also in a way of spontaneous speech. Previous attempts to build Amharic spontaneous speech recognizers are very limited in number. Bantegize [9] attempted to develop a spontaneous speech recognition system for the judiciary domain dictation application, and Adugna [10], explored domain independent spontaneous speech recognition systems which attain 55.46% words level correct recognition, and 39.86% accuracy.

The above mentioned studies conducted in the area are done using the phone-based speech unit. These has its own effect on the performance due to the reason that phone are highly context-sensitive than sub-word unit syllable. So we explore the study using CV-syllable as a recognition unit. For language modeling Adugna[10] uses bi-gram language model since the bi-gram has least context than the tri-gram; these also has its own effect on the performance of the recognizer.

Hence in this study an attempt is made to explore the extent to which the performance of Amharic Spontaneous Speech recognizer is improved using the tri-gram language model and cross-word tri-phone.

1.5 Research Questions

At the end of this study the following research questions are answered.

- To what extent changing the level of speech recognition unit from phone to syllable does improve the performance of Amharic spontaneous speech recognition?
- Is bi-gram or tri-gram language model performing better for Amharic spontaneous speech recognition?

1.6 Objective of the Study

The general and specific objectives of the study are the following:-

1.6.1 General Objective

The general objective of this study is to explore the effect of using cross-word tri-phone, changing the sub-word unit, and considering tri-gram language model so as to increase the performance of spontaneous speech recognizer for Amharic language.

1.6.2 Specific Objectives

To achieve the general objective of the study, this research attempts the following specific objectives.

- To build acoustic model using different sub-word units.
- To build dictionary using different sub-word units.
- To construct the language model using different n-gram models.
- To test the performance of the recognizer using different test set.

1.7 Scope and Limitation of the study

For this study we used speech corpus of Adugna [10] which estimated three hours and 10 minutes speech data a total of 2007 sentences for training and 104 sentences for testing. The test set used are two types: speaker independent and mixed test set. The pronunciation dictionary used for training and testing was canonical pronunciation dictionary prepared by taking syllable as a unit of recognition. Non speech events which are observed in our speech data are modeled by considering them as a word rather than considering them as a silence. The study doesn't consider different pronunciation for the same words which is uttered by different speakers since we prepare the canonical pronunciation dictionary. Because we are bounded by time and we didn't get the linguistic experts easily to work together for preparing alternative pronunciation dictionary.

The limitation of this study is that we use spontaneous speech corpus which is prepared and transcribed manually, since in this domain there are only two spontaneous speech corpus are available and both of them are prepared and transcribed manually. In addition to this the study doesn't consider different pronunciation for the same words which is uttered by different

speakers since we prepare the canonical pronunciation dictionary. Because we are bounded by time and we didn't get the linguistic experts easily to work together for preparing alternative pronunciation dictionary.

The approach we follow is stochastic approach and hidden markov model, because it has become so popular are the inherent statistical (mathematically precise) framework: the ease and availability of training algorithms for estimating the parameters of the models from finite training sets of speech data; the flexibility of the resulting recognition system in which one can easily change the size, type, or architecture of the models to suit particular words, sounds, and so forth; and the ease of implementation of the overall recognition system [27].

1.8 Significance of the study

In a day to day activity peoples communicate through speech. It is a focus area now a day to make the communication between people and machine through speech. The communication between people is using continuous (spontaneous) speech therefore peoples need to communicate with machine by spontaneous speech like they do with people; this study as a contribution towards developing a system that enable human being to communicate with a machine using spontaneous speech . Like other languages speech recognition, Amharic speech recognition is also very helpful for handicapped Amharic speakers that means for users who have difficulty in using their hands to type, but are able to speak clearly. In addition, blind users can use this kind of system since they have difficulty in using keyboard and mouse to command and control computers. Other group of users that can get benefit from this kind of system is people whose eyes and hands are busy in performing other task. In general, it can be said that such system is helpful for any people who can speak Amharic. This study is, therefore, a step towards the development of such a useful system [13]. This study has its own contribution on the applicability of speech recognition, and since the effectively broadening the application of speech recognition depends crucially on raising recognition performance for spontaneous speech. Knife has tried to state that the ultimate goals of ASR researches are speaker-independent continuous speech recognition on system. Since this study conducted on speaker-independent speech recognition it will have its significance for this ultimate goal of ASR. This study can be used as an input for future researches on Amharic speech recognition since there will be

recommendations from this study finding for future works in this area particularly in spontaneous speech recognition.

1.9 Research Methodology

1.9.1 Study design

The study follows an experimental research, a quantitative approach designed to discover the effect of presumed causes [23]. The key feature of this approach is that one thing is deliberately varied to see what happened to something else. Since we try to explore the effect of cross-word tri-phone, changing the sub word unit to CV-syllable, and tri-gram language model in order to explore such effect on the performance of the recognizer we select experimental research to go through.

1.9.2 Literature Review

The research which is conducted in every subject area should have link with the studies conducted in the similar area. We review different literatures (such as books, journal articles, conference papers, the Internet) from local as well as international literatures which help us to accomplish this study. To grasp wide concept about the speech recognition system we review international literatures, and to be specific in our domain we review the local literatures more related to this study.

1.9.3Corpus preparation

In this stage we used the corpus already prepared by Adugna[10]. For speech recognition system development we need three models (acoustic, lexical, and language models). In order to have these models we have to have audio and text data. Both the audio file and the transcription text is taken from Adugna[10] which applied for this specific study these audio and text data are applied according to their importance for where they are appropriate. Speech Data used for this study is taken from speech corpus prepared by Adugna[10].The audio data are collected from different online multimedia sources like YouTube and DireTube. These audio files are from different local mass media particularly from Sheger 102.1 FM radio, Ethiopian Broadcasting Corporation (EBC) and Ethiopian Broadcasting Service (EBS).

Totally the audio files are three hour and twenty minutes long, conversational speeches which are used both for training and for testing. They are not restricted to any domain rather they are general, and they are taken from an interview made between two and more people on different issues (domains) like sport, entertainment, politics, economy and others. These speeches are segmented and transcribed manually. Since these audio files can't be used for training and testing as they are collected from media, researcher [10] segmented in to sentences and transcribed manually. The data which is used for training is sentences from 36 total speakers and 17 of them are females and 19 of them are males, on average 56 sentences are uttered by each of the speakers both males and females. These sentences which are considered for training have 2007 number of utterances and these sentences (utterances) are constructed from 9460 number of unique words. The duration of all these speeches used for training is around 3 hours and 10 minutes. The additional data used is taken from Bantegize[9] which is judiciary domain based for dictation application . The speech corpus consists of 1556 sentences spoken by 16 people which estimated 1 hour and 30 minutes speech. The test data used for this study is also taken from Adugna [10]. The test data (Test) is constructed from, both the speakers which are involved in the training and not involved in the training. Test data have 14 total numbers of speakers and it involves utterances of 10 male speakers and 4 female speakers. The numbers of words in this test data are around 850unique words which are from 104 utterances (sentences) and it is around 10 minutes.

1.9.4Modeling techniques and tools

To come up with performance improvement for spontaneous speech recognition selection of approach and modeling tool is most important step for the process. The approach we follow in this study is pattern matching approach since it uses a well formulated mathematical framework, robustness and invariance to different speech vocabularies in users, feature sets, tern comparison algorithms and decision rules [27].

For the modeling techniques we select the Hidden Markov Model (HMM) as a modeling tool because it became the predominant techniques for speech recognition and a powerful statistical tool for modeling generative sequence that can be characterized by an underlying process generating an observable sequence. In addition to that the core of pattern matching speech recognition approach is a set of statistical models representing the various sounds of the language

to be recognized. Since speech has sequential structure and can be encoded as a sequence of spectral vectors, the Hidden Markov Model (HMM) provides a natural framework for constructing such models. For this study, HTK (Hidden Markov Model toolkit) will be employed as modeling tool because HTK is the most advanced and widely used system for modeling non-stationary data using HMM models [20].For language modeling we have used SRILM language modeling toolkit and for textmodification and transliteration we have also used Python code.

1.9.5. Evaluation procedure

In order to evaluate the speech recognizer we used the test corpus already prepared by [10].Which consists of 104 utterance collected from 14 speakers. Then we follow the evaluation procedure by measuring accuracy rate of the speech recognizer. The word error rate calculates miss recognitions at the word level: it compares the words outputted by the recognizer to those the user really spoke. Every error (substitution (S), insertion (I) or deletion (D)) is counted against the recognizer [1].

WER (Word Error Rate)= (Number of Substitution + Insertions + Deletions)/ Total number of words (N). Then, for measuring the performance of a speech recognition system, word recognition rate (WRR) is used by subtracting WER from 1.

1.10 Organization of the Thesis

This paper is divided into five chapters. Chapter one consists of introduction, statement of the problem, research question, objectives of the study, methodology followed in the course of the study and the scope of the study. In chapter two statistical speech recognition is reviewed the stochastic approach and their components are discussed. In addition related works are presented to show the gap that needs further research. Chapter three presents the characteristics of Amharic language, Amharic language writing system, and syllabification of Amharic language. Chapter four provides the experimentation, analysis, comparison and discussion about the result. Finally, conclusions and recommendations are given in chapter five.

CHAPTER TWO

LITERATURE REVIEW

2.1 Overview

Speech recognition is concerned with converting the speech waveform, an acoustic signal, into a sequence of words. Today's most practical approaches are based on a statistical modeling of the speech signal. This chapter focuses on the statistical methods used in state-of-the-art speaker-independent, continuous speech recognition. Some of the primary application areas of speech recognition technology are dictation, spoken language dialog and transcription systems for information retrieval from spoken documents [2]. The speech recognition problem we have to solve is, someone produces some speech and we have to have a system that automatically translates this speech into a written transcription. To solve this problem among different approaches we can use statistical approach. From a statistical point of view, speech is assumed to be generated by a language model which provides estimates of P(W) for all possible word strings W = (w1,w2,w3…wi), and an acoustic model represented by a probability density function p(O|W) encoding the message W in the signal O. The goal of speech recognition is generally defined as finding the most likely word sequence given the observed acoustic signal [2].

The main components of a generic statistical speech recognition system are shown in Figure2.1 along with the requisite knowledge sources (speech and textual training materials and the pronunciation lexicon) and the main training and decoding processes. The acoustic and language models resulting from the training procedure are used as knowledge sources during decoding, after feature analysis has been carried out from speech data by feature extraction (preprocessing). The rest of this chapter is devoted to discussing these main constituents and knowledge sources.

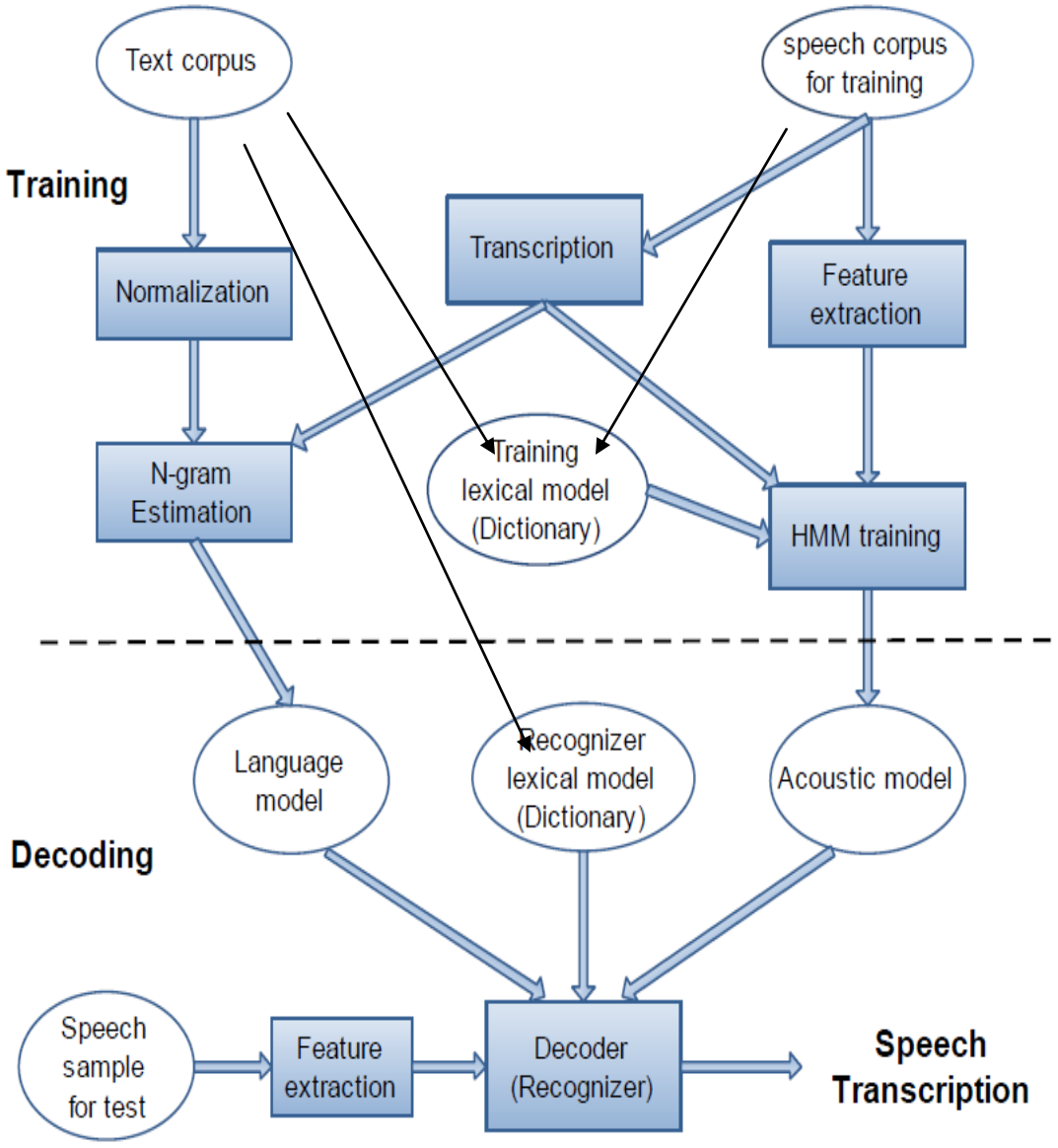

Figure2.1Architecture of an ASR system based on statistical approach, adapted from [8]

The literatures reveal that the components of speech recognition are categorized into three. The acoustic model, lexical model and language models are the well-known components of automatic speech recognition. However, the continuous speech waveform is first converted to a sequence of equally spaced discrete parameter vectors. The assumption is the speech vector is being stable, although this is not strictly true it is a reasonable approximation. Then by applying different parameters the extraction is done using feature extraction methods. Automatic Speech Recognition (ASR) systems have certain features which are composed of two parts: training and

decoding [14]. To train an ASR we need a set of training waveforms and their transcripts in order to build the acoustic, lexical, and language models. The acoustic model captures statistics on phoneme realization, the lexicon contains mappings of words to their underlying acoustic units, and the language model defines the likelihood of one word following another. The other part in automatic speech recognition is decoding considered as the heart of automatic speech recognition which search for the most likely word sequence given the observed features extracted from the speech signal. This is commonly referred to as decoding or recognizing the speech signal.

2.2 Signal Processing and Feature Extraction

Namrata has indicated that, every other component in a speech recognition system depends on two basic subsystems: signal processing and feature extraction [15]. The signal processing sub-system works on the speech signal to reduce the effects of the environment (e.g., clean vs. noisy speech), the effects of the channel (e.g., cellular/land-line phone versus microphone). The feature extraction sub-system parameterizes the speech waveform so that the relevant information (the information about the speech units) is enhanced and the non-relevant information (age-related effects, speaker information, and so on) is mitigated.

There are different methods that can be used to extract parameters of a speech [15], such as (1) production-based, (2) perception based, by simulating the effect that the speech signal has on the speech perception system (3)Signal based, method to describe the signal in terms of its fundamental components.

Regardless of the method employed to extract features from the speech signal, the features are usually extracted from short segments of the speech signal. This approach comes from the fact that most signal processing techniques assumes the vocal tract as stationary, but speech is non-stationary due to constant movement of the articulators during speech production. However, due to the physical limitations on the movement rate, a segment of speech sufficiently short can be considered equivalent to a stationary process. This approach is commonly known as short-time analysis, Signal based Analysis.

Perception-based analysis uses some aspects and behavior of the human auditory system to represent the speech signal. Given the human capability of decoding speech, the processing

performed by the auditory system can tell us the type of information and how it should be extracted to decode the message in the signal.

2.2.1 Feature Extraction

Features extraction in ASR is the computation of a sequence of feature vectors which provides a compact representation of the given speech signal [15]. It is mainly performed in three main stages. The first stage is called the speech analysis or the acoustic front end, which performs spectra temporal analysis of the speech signal and generates raw features describing the envelope of the power spectrum of short speech intervals. The second stage compiles an extended feature vector composed of static and dynamic features. Finally, the last stage transforms these extended feature vectors into more compact and robust vectors that are then supplied to the recognizer.

Feature Extraction is the most important part of speech recognition since it plays an important role to separate one speech from other [7]. Because every speech has different individual characteristics embedded in utterances. These characteristics can be extracted from a wide range of feature extraction techniques proposed and successfully exploited for speech recognition task. Extracted feature should meet some criteria while dealing with the speech signal such as [15]:

- It should be easy to measure extracted speech features.
- It should not be susceptible to imitation.
- It should show little fluctuation from one speaking environment to another.
- It should be stable over time.
- It should occur frequently and naturally in speech.

The purpose of feature extraction stage is to extract the speaker-specific information in the form of feature vectors [7]. In addition, it is used to represent the speech wave forms in more compact manner. There are different techniques (methods) in order to extract feature from speech [15].

2.2.2 Production based analysis

The speech production process can be described by a combination of a source of sound energy modulated by a transfer (filter) function. This theory of the speech production process is usually referred to as the source -filter theory of speech production . The transfer function is determined by the shape of the vocal tract, and it can be modeled as a linear filter. However, the transfer function is changing over time to produce different sounds. The source can be classified into two types [15]. The first one is which is responsible for the production of voiced sounds (e.g., vowels, semivowels, and voiced consonants). This source can be modeled as a train of pulses. The second one is related to unvoiced excitation. In this type, this source can be modeled as a random signal.

2.2.3 Perception-based Analysis

Perception-based analysis uses some aspects and behavior of the human auditory system to represent the speech signal. Given the human capability of decoding speech, the processing performed by the auditory system can tell us the type of information and how it should be extracted to decode the message in the signal. Two methods that have been successfully used in speech recognition from this method of analysis are; Mel-Frequency Cepstrum Coefficients (MFCC) and Perceptual Linear Prediction (PLP) [15].

Mel-Frequency Cepstrum Coefficients (MFCC)

Mel Frequency Cepstrum Coefficients (MFCC) is the most prevalent and dominant method used to extract spectral features is calculating Mel Frequency Cepstral Coefficients (MFCC). MFCCs are one of the most popular feature extraction techniques used in speech recognition based on frequency domain using the Mel scale which is based on the human ear scale. MFCCs being considered as frequency domain features are much more accurate than time domain features.

The Mel-Frequency Cepstrum Coefficients is a speech representation that exploits the nonlinear frequency scaling property of the auditory system. This method warps the linear spectrum into a nonlinear frequency scale, called Mel. The Mel-scale attempts to model the sensitivity of the human ear and it can be approximated by the following formula [15]:

$$B(f) = 1125 \ln\left(1 + \frac{f}{700}\right), \qquad 2.1$$

For frequency f, the scale is close to linear for frequencies below 1 kHz and is close to logarithmic for frequencies above 1 kHz [21]. MFCCs which are implemented for this study are often used in many other speech recognition systems.

Body Linear Predictive Codes (LPC)

Body Linear Predictive Codes (LPC) is desirable to compress signal for efficient transmission and storage. Digital signal is compressed before transmission for efficient utilization of channels on wireless media. For medium or low bit rate coder, LPC is most widely used [13]. The LPC calculates a power spectrum of the signal. It is used for formant analysis [14]. LPC is one of the most powerful speech analysis techniques and it has gained popularity as a formant estimation technique.

Perceptual Linear prediction (PLP)

The Perceptual Linear Prediction PLP model developed by Hermansky. PLP models the human speech based on the concept of psychophysics of hearing [15]. PLP discards irrelevant information of the speech and thus improves speech recognition rate. PLP is identical to LPC except that its spectral characteristics have been transformed to match characteristics of human auditory system.

2.3 Acoustic models

The acoustic model determines the sound that will be produced when a given string of words is uttered. Thus, for all possible combinations of word strings and observation sequences theProbability,*P* (O |W) must be available [1].

$$W = argmax\, P(W \mid O) \quad \text{.....2.2}$$

This number of combinations is just too large to permit a lookup; in the case of continuous speech it's even infinite. The total process modeled involves the way the speaker pronounces the words of W, the ambience (room) noise, the microphone placement and characteristics and the acoustic processing performed by the signal processing front end.

Unfortunately, unless there is some limit on the duration of the utterances and a limited number of observation symbols, this equation is not directly computable since the number of possible observation sequences is totally infinite. But Bayes formula gives:

$$P(W \mid O) = \left(P(W)P(O \mid W)\right)/(P(O)) \quad \text{2.3}$$

From the above formula, *P*(W), is called the language model, which is the probability that the word string W will be uttered and *P*(O|W) is the probability that when word string W is uttered the acoustic evidence O will be observed, which is called the acoustic model. The probability *P*(O) is usually not known but for a given utterance it is of course just a normalizing constant and can be ignored. Thus to find a solution to formula (2.2) we have to find a solution to:

$$W = argmaxP(W)P(O \mid W) \quad \text{2.4}$$

The most frequently used model these days is the Hidden Markov model but other models are possible, for instance artificial neural networks [8].

Acoustic model is the main component for an ASR and it accounts for most of the computational load and performance of the system. It is used to link the observed features of the speech signals with the expected phonetics of the hypothesis sentence and used for detecting the spoken phoneme. Its creation involves the use of audio recordings of speech and their text scripts and then compiling them into a statistical representation of sounds which make up words. Acoustic models represent sub-word units, such as phonemes, as a finite-state machine in which states model spectral structure and transitions model temporal structure.

2.3.1 Hidden Markov Model (HMM)

The core of pattern matching speech recognition approach is a set of statistical models representing the various sounds of the language to be recognized. Since speech has sequential structure and can be encoded as a sequence of spectral vectors, the hidden Markov model (HMM) provides a natural framework for constructing such models.

HMM is a Markov chain plus emission probability function for each state. In the Markov model each state corresponds to one observable event. But this model is too restrictive, for a large number of observations the size of the model explodes, and the case where the range of observations is continuous is not covered at all.

As described by Jurafsky, et.al [16] an HMM is specified by a set of states Q, a set of transition probabilities A, a HMM set of observation likelihoods B, a defined start state and end state(s), and a set of observation symbols O, which is not drawn from the same alphabet as the state see

Q:

A Hidden Markov model can be defined by the following parameter [27]:

- *S*= {s1,s2,...s*N*}: A set of states (usually indicated by *i, j) is* a state that the model is in at a particular point in time t. it will be indicated by st, thus *st = i* means that the model is in state *i* at time *t*.
- A = a11 a12 ... a*ij*: A transition probability A, each aij representing the probability of moving from state i to state j.
- O= o1 o2 … oN : A set of observations, each one drawn from a vocabulary V= v1, v2…vv
- B = bi (ot) : A set of observation likelihoods: also called emission probabilities, each expressing the probability of an observation ot being generated from a state i.
- $\pi = \pi 1, \pi 2, \ldots, \pi N$: An initial probability distribution over states :πi is the probability that *si*is *s* starting state.
- $\lambda = (A,B,\pi)$: Full HMM

HMM Problems and Their Solution

HMM three basic problems are Evaluation, Decoding and Training [21]. The next topics will discuss these three problems and their solution.

Problem1 (Computing likelihood): Given an HMM $\lambda = (A, B, \pi)$ and an observation sequence O, determine the likelihood $P(O|\lambda)$?

Problem2 (Decoding): Given an observation sequence O and an HMM λ = (A, B, π), discover the best hidden state sequence Q?

Problem3 (Learning): Given an observation sequence O and the set of states in the HMM, learn the HMM parameters A and B.

Solution to Problem 1 (computing likelihood): The Forward Algorithm

The forward algorithm is a kind of dynamic programming algorithm, an algorithm that uses a table to store intermediate values as it builds up the probability of the observation sequence. The forward algorithm computes the observation probability by summing over the probabilities of all possible hidden state paths that could generate the observation sequence, but it does so efficiently by implicitly folding each of these paths into a single forward frame

Each cell of the forward algorithm frame αt (j) represents the probability of being in state *j* after seeing the first *t* observations, given the model λ. The value of each cell αt (j) is computed by summing over the probabilities of every path that could lead us to this cell. Formally, each cell expresses the following probability [27]:

αt(j)= P(o1,o2,…ot,qt = si| λ)……..…..2.5

We compute this probability by summing over the extensions of all the paths that lead to the current cell. For a given state siat time *t*, the value *αt (j)* is computed as:

$$\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} \right] b_j(o_t) \quad \text{..... 2.6}$$

The three factors that are multiplied equation 2.6 for extending the previous paths to compute the Viterbi probability at time t are: αt −1 (i).The previous forward path probability from the previous time step αij. The transition probability from previous state qi to current state qjbj(ot) The state observation likelihood of the observation symbol otgiven the current state j. We can define the forward algorithm using a statement of the definitional recursion [27]:

- ✓ Initialize

$$\alpha_1(i) = \pi_i b_i(o_1) 1 \le i \le N \quad \text{.....2.7}$$

- ✓ Recursion (since states 0 and N are non-emitting)

$$\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} \right] b_j(o_t) \;\; 2 \le t \le T, 1 \le j \le N \quad \text{.....2.8}$$

- ✓ Termination

$$P(O \mid \lambda) = \sum_{i=1}^{N} \alpha_T(i) \quad \text{.....2.9}$$

Solution to HMM Problem2 (Decoding): Viterbi algorithm

Decoding problem deals with, given a model and an observation sequence, finding the most likely or optimal state sequence in the model that produced the observation sequence. Since the state sequence is hidden in an HMM. Thus, to solve the problem it is possible to produce the state sequence that has the highest probability of being taken while generating the observation sequence. To do this we can use viterbi algorithm, which is a modification of forward algorithm. Instead of summing probabilities that came together as in the forward algorithm, in viterbi we need to choose and remember the maximum probability. The viterbi algorithm has one component that the forward algorithm does not have: back pointers. This is because while the forward algorithm needs to produce observation likelihood, the Viterbi algorithm produces a probability and also the most likely state sequence [1]. We compute this best state sequence by keeping track of the path of hidden states that led to each state.

We want to find the state sequence Q=q1…qT, such that [27]:

$$Q = \arg\max_{Q'} P(Q' \mid O, \lambda) \quad \text{.....2.10}$$

Similar to computing the forward probabilities, but instead of summing over transitions from incoming states, compute the maximum:

$$\delta_t(j) = (\max_{1 \le i \le N} \delta_{t-1}(i) a_{ij}) b_j(o_t) \quad \text{.....2.11}$$

Solution to Problem3: The Forward – Backward Algorithm (Baum-*Welch algorithm)*

The third problem of HMM is the learning (training) problem in which, given the model and an observation sequence, we attempt to adjust the model parameters to maximize the probability of generating the observation sequence. Rabiner and Juang [1] supposed this problem is the most difficult problem since there is no known analytical method to solve for the model parameters that maximizes the probability of the observation sequence. An iterative procedure is used to solve this problem. One iterative procedure that is used to solve this problem is the forward – backward algorithm, which is also called Baum Welch algorithm. Using an initial parameter instantiation, the forward-backward algorithm iteratively re-estimates the parameters and improves the probability that given observations is generated by the new parameters.

Here there are three parameters need to be re-estimated [1]:

- Initial state distribution:π_i .
- Transition probabilities: $a_{i,j}$.
- Emission probabilities: $b_i(ot)$

Re-estimating the transition probabilities

Here we have to solve, as noted by [1], what is the probability of being in state s_i at time t and going to state s_j, given the current model and parameters?

$$\xi_t(i,j) = P(q_t = s_i, q_{t+1} = s_j \mid O, \lambda) \quad \text{.....2.18}$$

Let $\xi(i,j)$ be a probability of being in state *i* at time *t* and at state *j* at time *t+1*, given λ and O;

$$\xi(i,j) = \frac{\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}{P(O \mid \lambda)}$$

$$= \frac{\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)} \quad \text{.....2.19}$$

The perception behind the re-estimation equation for transition probabilities is:

$$\hat{a}_{i,j} = \frac{\text{expected number of transitions from state } s_i \text{ to state } s_j}{\text{expected number of transitions from state } s_i} \;;$$

$$\hat{a}_{i,j} = \frac{\sum_{t=1}^{T-1}\xi_t(i,j)}{\sum_{t=1}^{T-1}\sum_{j'=1}^{N}\xi_t(i,j')} \quad \text{.....2.20}$$

The above equation can be modified as:

$$\hat{a}_{i,j} = \frac{\sum_{t=1}^{T-1}\xi_t(i,j)}{\sum_{t=1}^{T-1}\gamma_t(i)} \quad \text{.....2.21}$$

Re-estimating Initial state probability

Initial state distribution is the probability that si is a starting state.

Re-estimation is π: 1 at time s state in times of number expected.

$$\hat{\pi}_i = \gamma_1(i) \quad \text{.....2.22}$$

Re-estimation of Emission probabilities

$$\hat{b}_i(k) = \frac{\text{expected number of times in state } s_i \text{ and observe symbol } v_k}{\text{expected number of times in state } s_i}$$

$$\hat{b}_i(k) = \frac{\sum_{t=1}^{T} \delta(o_t, v_k)\gamma_t(i)}{\sum_{t=1}^{T} \gamma_t(i)} \quad \text{.....2.23}$$

2.4 Lexical Models

Lexicon is developed to provide the pronunciation of each word in a given language [3]. Through lexical model, various combinations of phones are defined to give valid words for the recognition. The type of speech unit determines the pronunciation dictionary to be prepared .i.e. the lexicon prepared for phones are not the same as for the speech unit syllable. A pronunciation dictionary can be classified as a canonical or alternative on the basis of the pronunciations it includes. A canonical pronunciation dictionary includes only the standard phone or other sub-word sequence assumed to be pronounced in read speech. It does not consider pronunciation variations such as speaker variability, dialect, or co-articulation in conversational speech. On the other hand, an alternative pronunciation dictionary uses the actual phone or other sub-word sequences pronounced in speech. In an alternative pronunciation dictionary, various pronunciation variations can be included.

2.5 Language Models

Language modeling is a task assigning a probability to a given sequence of words [8]. It provides word-level structure for a language and applicable for many speech and natural language application such as automatic speech recognition (ASR), statistical machine translation (SMT) and optical character recognition (OCR). By using Language modeling automatic speech recognition (ASR) benefited in two ways. It reduces the search space for searching and providing contextual information [8].

Since language model ignore unlikely candidates, and thus the search problem become feasible. With regard to the second concept in speech, there is a concept homonym meaning their pronunciation is exactly the same. Language model provide us contextual information to distinguish homonyms most of the time we can come up with a clear distinction between homonyms by applying language modeling [8]. To assign probability to given input word string the most popular language model is n-gram model. Based on the number of word used for conditioning the n-gram language model is classified into three namely uni- gram, bi-gram and tri-gram. If one word is used for conditioning to determine the probability of word it is called uni-gram, in other case if we are considering two words for conditioning to determine the probability of word it is bi-gram and if we are considering three words for conditioning to determine the probability it is tri-gram. Language model predicts the next set of words, and controls which models are hypothesized. It is the single largest component consists of parameters and developed for detecting the connections between the words in a sentence with the help of pronunciation dictionary [4].

ASR systems utilize *n*-gram language models to guide the search for correct word sequence by predicting the likelihood of the *n*th word on the basis of the $n-1$ preceding words .N-gram language model are easy to build as well as we need simple count of the word n-gram event from the training set. But there is a problem in n-gram model for the unseen word in the training set the probability is zero this affect s the other word which appear in the training set is also zero, since the product of the two is zero. In order to solve the problem of n-gram model there is an introduction of several smoothing techniques among them several of them work well in ASR applications. “The fundamental idea of smoothing technique is to reserve (subtract) some small probability mass from the relative frequency estimate of the probabilities of seen event, and to

redistribute this probability to unseen events" [8]. Smoothing methods differs according to how much is subtracted out (discounting) and how it's redistributed (back-off).

2.6 Speech Units in Automatic Speech Recognition

A crucial issue for acoustic modeling is the selection of the speech units that represent the acoustic and linguistic information for the language. Speech is represented by using the various form of speech unit in automatic speech recognition. Ranging from smallest unit phones to the largest which is the word. Word is the natural unit of speech because ultimately it is word that one is trying to recognize. Moreover, word units have their acoustic representation well defined. Acoustic variability occurs mainly in the beginning and end of a word i.e. at word boundaries.

However, using word as a speech unit in large vocabulary continuous speech recognition (LVCSR) introduces several serious problems. Since each word has to be trained individually and there cannot be any sharing of parameters among words, one has to have a very large training set so that all words in the vocabulary are adequately trained. The second problem lies with memory requirement which grows linearly with the number of words. Hence word models are not practical for large vocabulary continues speech recognition (LVCSR) systems. But word models have been successfully used for limited vocabulary ASR [17].

A smaller or sub-word unit is the next choice because one has to have sharing of parameters across models in order to save computing resources. The most popular sub-word units are the phones, as it is a well-known fact that the same phone in different words has different realizations. This is because the articulators cannot move from one position to another instantaneously. Hence the realization of a phone is strongly affected by its adjacent phones or in other words, phones are highly context dependent [17]. Some phones are aspirated when they occur in the beginning of a word and the same phones are not aspirated when they occur at the end of a word. Therefore the acoustic variability of basic phonetic units due to context is sufficiently large and not well understood in many languages. This problem of over-generalization can be overcome by employing phone-in-context. A context refers to immediate left and/or right neighboring phones. Phone-in-context can be either a left-context phone or a right-context phone or both. The third category is also known as tri-phone which includes both the left and right contexts. If two phones have the same identity but different left or right contexts, then they are considered as different tri-phones. Tri-phone models are powerful sub-

word models because they account for the left and right phonetic contexts. Tri-phones have been enormously successful in acoustic modeling of LVCSR systems [17]. Compared to word-model and phone-model, tri-phone model reduced word error rate (WER) by more than 50% [17]. However there are problems with tri-phones also. For any language, there are large numbers of tri-phones in the training set. This leads to memory wastage since a model is created for a tri-phone even if, that tri-phone is observed only once in the training set. A syllable is a larger unit than a phone since it encompasses two or more phone clusters [17]. These phone clusters account for the severe contextual effects. A syllable is composed of three parts: the onset, the nucleus and the coda or rhyme. The nucleus or central part does not have any contextual dependencies while the onset and the coda are still susceptible to some contextual effects. Syllables have long unit and they have the least context sensitive [18].The advantage of using syllables as a unit of training is that pronunciation variation is trained right into the acoustic model and does not need to be modeled separately in the dictionary. Syllable models also automatically capture co-articulation effects [19].

2.7 Automatic speech recognition tools

The following tools are freely available to apply and use them in automatic speech recognition system. They trigger our task by performing within a short period of time.

Hidden Markov Model Toolkit (HTK):- HTK is the most advanced and widely used system for modeling non-stationary data using HMM models [20]. It is free for educational or academicals purposes and can be downloaded after a registration. The main advantages of the HTK are: it is a complex system that covers all development phases of a recognition system, system is regularly updated to catch up with the latest advances in the field of recognition, and it is well documented both theoretically and practically used to model any time series data. HTK is primarily designed for building HMM-based speech processing tools, in particular recognizers. This particular study also used HTK tools because of the above mentioned reasons. Two major processing stages involved. Firstly, the HTK training tool is used to estimate the parameters of a set of HMMs using training utterance and their associated transcription. Secondly, unknown utterance is transcribed using HTK recognition tools. The HTK tools broadly categorized into four parts data preparation Tools, Training tool, recognition tool, and analysis tool.

i. Data preparation tools

In order to build a set of HMMs, a set of speech data files and their associated transcriptions are required [21]. If the speech needs to be recorded, then the tool HSLab can be used both to record the speech and to manually annotate it with any required transcriptions. Although all HTK tools can parameterize waveforms on-the-fly, in practice it is usually better to parameterize the data just once. The tool HCopy is used for this. As the name suggests, Hcopy is used to copy one or more source files to an output file. Normally, HCopy copies the whole file, but a variety of mechanisms are provided for extracting segments of files and concatenating files. By setting the appropriate configuration variables, all input files can be converted to parametric form as they are read-in. Thus, simply copying each file in this manner performs required encoding. The tool 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. Transcriptions will also need preparing. Typically the labels used in the original source transcriptions will not be exactly as required, for example, because of differences in the phone sets used. Also, HMM training might require the labels to be context-dependent. The tool HLEd is a script-driven label editor which is designed to make the required transformations to label files. HLEd can also output files to a single Master Label File MLF which is usually more convenient for subsequent processing. Finally on data preparation, HLStats can gather and display statistics on label files and where required, HQuant can be used to build a VQ codebook in preparation for building discrete probability HMM system

ii. Training Tools

The HTK training tools are used to estimate the parameters of a set of HMMs using training utterances and their associated transcription [21]. The second step of system building is to define the topology required for each HMM by writing a prototype definition. HTK allows HMMs to be built with any desired topology. HMM definitions can be stored externally as simple text files and hence it is possible to edit them with any convenient text editor. Alternatively, the standard HTK distribution includes a number of example HMM prototypes and a script to generate the most common topologies automatically [21]. 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 actual parameters will be computed later by the training tools. 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.

iii. Recognition Tools

HTK provides a recognition tool called HVite that allows recognition using language models and lattices [21]. HLRecsore is a tool that allows lattices generated using Hvite (or HDecode) to be manipulated for example to apply a more complex language model. An additional recognizer is also available as an extension to HTK Hdecode[21].

iv. Analysis tool

A tool called HResults compares recognition results with original transcriptions [21]. It uses dynamic programming to align the two transcriptions and then counts substitution, deletion and insertion errors. Options are provided to ensure that the algorithms and output formats used by HResults are compatible with those used by the US National Institute of Standards and Technology (NIST). As well as global performance measures, HResults can also provide speaker-by-speaker Break downs, confusion matrices and time-aligned transcriptions. For word spotting applications, it can also compute Figure of Merit (FOM) scores and Receiver Operating Curve (ROC) information. There are three possible types of errors that can be analyzed which are made during recognition: An *insertion error* occurs when the ASR system generates a word that does not correspond to any word in the reference transcript. A *deletion error* occurs when the reference transcript contains a word that has no corresponding word in the ASR hypothesis. A *substitution error* occurs when the corresponding word in the ASR transcript is different than that of the reference transcript. Once it founds the optimal alignment, HRESULTS calculates the number of substitution errors (*S*), deletion errors (*D*) and insertion errors (*I*) [21]; where *N* is the total number of labels in the reference transcriptions.

2.8 Applications of Speech Recognition

Speech recognition is often used as the front-end for many natural language processing (NLP) applications. Some of these applications include machine translation, information retrieval and extraction, voice dialing, call routing, speech synthesis/recognition, data entry, dictation, control, etc. Thus, much research work has been done to improve the speech recognition and the related NLP applications [22]. The crucial impact of speech recognition lies whether one can fully integrate the enabling technologies with applications. How to successfully integrate speech into applications often depends on the nature of the user interface and application. An elegant user interface requires carefully considering the particular user group of the application and delivering an application that works effectively and economically. One exclusive challenge in speech-recognition applications is that speech recognition (as well as understanding) is unsatisfactory.

It is therefore essential that applications make use of necessary interactive error handling techniques to minimize the impact of these errors. There are many reasons why many researchers are being committed to develop systems with speech recognition interfaces [22]. One of the justifications is its naturalness. Speech is the most natural and universal method of communication between people. The aim of speech recognition is to extend that communication modality to interaction with computers. Speech recognition is being programmed into computer software to provide a more natural interface for much of our thinking about speech recognition has been focused on it use as an interface in human-machine interaction mostly for information access and extraction.

With increase in cellular phone use and dependence on networked information resource, and rapid access to information become an increasingly important economic factor, telephone access to data and telephone transactions will no doubt rise dramatically. There is a growing interest, however, in viewing speech not just as a means to access information, but as itself, a source of information. We can envision a great information revolution on par with the development of writing systems, if we can successfully meet the challenges of speech both as a medium for information access and as itself a source of information.

Speech is still the means of communication used first and foremost by humans, and only a small percentage of human communication is written. It is also used by Novice computer users, including busy professionals who cannot take time to learn complex keyboard commands. It reduces training time while increasing ease-of-use. According to [16], five general areas of speech recognition applications have been identified: assistive technology, command and control, telecommunications, data entry and retrieval, and education. Speech recognition applications can be used as an assistive technology the primary purpose of which is aiding persons with severe disabilities. Speech recognition systems enable handicapped and visually impaired individuals control their environment by providing a voice interface to a computer, telephone, fax machine, etc. [16].Command and control refers to the control of machines via computers which are themselves instructed through speech. Such applications are useful in industrial, military and health care environments. Speech recognition is also used in Telecommunications. Moreover, global telecommunication companies and researchers are working very hard on automated spoken language translation systems. The service, when realized, enables people speaking different languages to communicate easily and readily without language and cultural barriers. Speech recognition is also considered vital in data entry and retrieval process when "hands-busy, eyes-busy" or "remoteness" conditions prevail [27]. Data entry and retrieval refers to storage and retrieval of data by voice. One of the ultimate goals of speech recognition is to create a human-like listening typewriter or automated secretary Dictation systems are one successful speech recognition services that enable user to dictate directly into the computer. Speech recognition can also play a role in Computer Assisted Instruction (CAI) for disabled persons who are unable to use the keyboard or a pointing device. Moreover, properly designed Computer Assisted Instruction (CAI) systems provide an opportunity for anyone to acquire new skills or upgrade old ones in a cost-effective and efficient manner. A computer with speech recognition ability is a perfect language tutor – patient, positive, and cheerful .These is very motivating application areas in which many subsequent researches could be conducted to adopt these technologies into our country with our own languages.

2.9 Performance evaluation of Speech Recognition System

Speed and accuracy are the two factors that specify performance of speech recognition systems. Accuracy may be measured in terms of performance accuracy which is usually rated with word error rate (WER), whereas speed is measured with the real time factor. To calculate speed of an ASR system real time factor (RTF) is the parameter used. If an input of duration 'I' takes time 'P' to process, then real time factor is mathematically defined as [2],

RTF=P/I

The word error rate calculates miss recognitions at the word level: it compares the words outputted by the recognizer to those the user really spoke. Every error (substitution, insertion or deletion) is counted against the recognizer [8].

WER = Number of Substitution + Insertions + Deletions/ Total number of words.

WER= (S+D+I/N). When considering the performance of a speech recognition system, at that time word recognition rate (WRR) is used instead, WRR = 1-WER = (N-S-D-I/N) = (H-I/N)

2.10 Related works

In this section we present international works on spontaneous speech recognition that are done using different techniques, methods and approach specially for handling non-speech event like filled pause, hesitation and word repetition in order to increase the performance of spontaneous speech. In addition, more related literatures on spontaneous speech are for local languages are also reviewed. Spontaneous speech recognizer for Japanese developed by Furui et.al [5] using two different Corpus; One "Corpus of Spontaneous Japanese (CSJ)" speech: A part of the corpus completed by the end of December 2000, consisting of 610 presentations (approximately 1.5Mwordsof Transcriptions), is used. Web corpus: Transcribed presentations consisting of approximately 76k sentences with 2M words have been collected from the World Wide Web used. In this experiment for the evaluation of recognizer performance, 4.4 hours of presentation speech uttered by 10 male speakers is used as a test set of speech recognition. Two acoustic model of tied-state tri phone have been made, the first acoustic model made using338 presentations in the CSJ uttered by male speakers (approximately 59 hours).

The speakers have no overlap with those in the test set. The second acoustic model made using approximately 40-hours of read speech uttered by many speakers. Both acoustic models have 16 Gaussian mixtures in each state. Using data from the above corpuses (CSJ and web corpus), two languages models have been constructed. One language model, made using the 610 presentations in the CSJ and the speakers have no overlap with those of the test set. Another language model, made using the text of web corpus. Each model consists of bigrams and reverse trigrams with backing-off. Their vocabulary sizes are 30k word [1].Acoustic model developed using CSJ and acoustic model developed using read speech evaluated using test data set [5]. The language model using the 610 presentations in the CSJ is used and they achieved 53 % and 65.3 % word accuracy respectively. This result indicates that it is crucial to make acoustic models and language models from a spontaneous speech corpus to adequately recognize spontaneous speech.

The acoustic model made using 338 presentations in the CSJ uttered by male speakers approximately 59 hours gives much better results than the acoustic model made using approximately 40 hours of read speech. From this result they suggested that acoustic models made from CSJ have better coverage of tri phones (since it is large in size) and better matching of acoustic characteristics corresponding to the speaking style and also have well matching of recording conditions with the test set. Furui, et.al [5] developed a recognizer using a large-scale spontaneous speech database "Corpus of Spontaneous Japanese (CSJ)". The acoustic model made using training data of 510 hours long and by using the whole training data set around 6.84 M words for language modeling, the best Word Error Rate of 25.3% obtained.

Spontaneous speech recognition for Amharic language using HMM is done by Adugna [10]. In this thesis work the speech unit selected is phone using Hidden Markov Model (HMM) and small vocabulary size is used. The recognizer has been tested using test data consisting of 104 utterances which is composed of around 850 unique words. The sentences (utterances) are from15 speakers which five of them are females and 10 of them are males. The recognizer is tested using three test data set speakers dependent, mixed group, and speaker independent and evaluated accordingly. With the speaker dependent 41.0% accuracy is achieved, with the mixed category the accuracy 39.1% achieved, and finally the recognizer is tested with speaker independent 25.1% s achieved. In order to improve the recognizer performance increasing the Gaussian mixture events modeled was, 55.46% percentage of words recognized and 39.86 %

percentage of word accuracy. From this result we can see that the percentage of words recognized is higher than the percentage of word accuracy. Percentage of word accuracy analysis includes insertion error but percentage of word recognized does not include insertion error. Therefore from this analysis we can say that there is insertion error which is decreasing the accuracy of the recognizer.

The literatures shows that spontaneous speech is highly affected by co-articulation effect and to improve the strong co-articulation between words, using a compound word tokens or the unit which is not context dependent like the syllable, to model the phoneme-deletion phenomenon, allowing a shorter path in HMM topology using 5-state model, to decrease the confusion caused by pause-fillers, laughter or other non-speech events, explicitly modeling the pause-fillers in the language model, assigned longer duration for them and trained a new model for laughter. Increasing the Gaussian mixture (GMM), and using tri-gram language modeling with smoothing techniques also important. All of these methods are directly derived from analysis of the physical phenomenon of spontaneous speech and the resulting precision is quite satisfactory. They're very good examples for telling that in dealing with non-stationary speech signal, knowledge of its particular behaviors is very important and helpful. Bantegize [9] developed recognizer using 90 min speech database in a judicial domain containing 1550 utterances for training the acoustic model. Language model developed using 30MB text data and different smoothing techniques. 68 sentences from the same domain used for testing. Using context dependent acoustic model of 8 Gaussian mixtures and a tri-gram language model with absolute discounting smoothing, result obtained were 50 % word accuracy and 53 % WER. Based on experimental result, Bantegize[9] if data size is less in number, the data used for training and testing from the same domain performs better than data which is not from specified domain.

This study has a great contribution towards improving the performance of the spontaneous speech recognizer for Amharic language using the parameter tuning techniques, tri-gram language model, and by changing the sub word unit to CV-syllable.

CHAPTER THREE

THE AMHARIC LANGUAGE

3.1 Overview of the Amharic language

This chapter gives an overview of the Amharic language in relation to speech recognition application. Like any other language Amharic language has its own syntax and structure. In addition to this it has a wide regional coverage and rich in morphology [23]. Ethiopia is a linguistically diverse country where more than 80 languages are used in day-to- day communication among people. Although many languages are spoken in Ethiopia, Amharic is dominant in that it is spoken as a mother tongue by a substantial segment of the population and it is the most commonly learned second language throughout the country. The language is the official working language of the federal government of the country, in different regions. The 2007 census of Ethiopian central statistical authority [30], Amharic is used for more than 32.7% of the population as first and second language.

Amharic is the Semitic language that has the second large number of speakers after Arabic. Most of the Semitic languages are technologically unflavored [11]. Amharic is one of these languages that are looking for technological considerations of researchers and developers in the area of natural language processing (NLP). Automatic Speech Recognition (ASR) is one of the major areas of NLP that is understudied in Amharic. Being a Semitic Language of the Afro-Asiatic Language Group, this language is related to Hebrew, Arabic, and Syrian. Unlike Arabic, Hebrew, or Syrian, Amharic language is written from left to right. It is one of the most widely spoken languages in Ethiopia. It has its own script that is borrowed from Geez, another Ethiopian Semitic language [23]. The script is believed to have originated from the South Sabean script. It is a syllabary writing system where each character represents an open CV syllable, i.e., a combination of a consonant followed by a vowel. According to [23] the foundation for any languages is sounds. Sounds make phonemes the phonemes in turn make up words, the words then sentences and so on. Sentences fully describe a thought. As expression of thoughts, a language is collection of sentences. People express their ideas in a given language through speaking. It is only few years ago that people started writing. This is a short time, even today, many nations and nationalities use speech to exchange ideas. The situation in our country is not different from this.

3.2 Amharic language script

The Ethiopic (Ge'ez) script was developed as the writing system of the Ge'ez language, a Semitic language spoken in Ethiopia and Eritrea until the 10^{th} to the 12^{th} centuries. The original Ge'ez script was an Abjad - vowels were not written - but the current script is classified as an Abugida. Each symbol represents a CV-syllable, but vowels are not inherent in the consonant.

According to [23] the original Ethiopic script contained 182characters, although the basic (unmarked) consonants number only 26. The script has since been extended for other languages and now contains over 500 symbols. Some of the new symbols represent phonological processes such as palatalization, pharyngealization, and labialization. The unmarked set is known as the first order or first form. Each of the first order consonants can be combined with one of six vowels, to produce a syllograph. The resulting sets of syllographs are known as the second, third, fourth, fifth, sixth and seventh orders. Although the language ceased to be used in vernacular speech (it now serves aliturgical function only), the script is still widely used for writing the Ethiopian and Eritrean Semitic languages such as Tigré, Amharic and Tigrigna.

Amharic script, unlike other Semitic scripts, is written from left to right and found in five dialectical variations (Addis Ababa, Gojjam, Gonder, Wollo, and Menz) spoken in different regions of the country.

3.3 Characteristics of Amharic language

Amharic uses thirty-three consonant classes and seven vowels from the Ethiopic alphabet, of which twenty seven have unique sounds, these being characterized in terms of their sound creation (labial, dental, palatal, velar, and glottal) and/or their graphic symbols [23]. Each consonant has seven forms created by fusion of a consonant and a vowel-fused form of vocalic marker. As citedon [11], there are a few other characters, called labiovelars that are created by rounding the lips when voicing consonants.

The word units of Amharic are: phoneme, morpheme, root, stem, and word [11]. A phoneme represents a basic sound or unit of sound. Every glyph or consonant form is a phoneme or unit of word. A phoneme or collection of phonemes forms a morpheme, which is the smallest meaningful unit in word [23]. A morpheme can be free or bound, where a free morpheme can stand as a word on its own whereas a bond morpheme cannot. An Amharic root is a sequence of

consonants, and is the basis for the derivation of verbs; a stem, on the other hand, is a consonant or consonant-vowel sequence. A stem can be free or bond: a free stem can stand as a word on its own whereas a bound stem has a bound morpheme affixed to it. A collection of phonemes or sounds creates a word, which can be as simple as a single morpheme or contain several of them. Since to build large vocabulary speech recognition system it is recommended to use the word units of that language. This research work is proposed to work on the syllable units of the Amharic language. The smallest unit of one language is its phone. Amharic has its own phonemes. As same with other Semitic languages, Amharic has its own characterizing phonetic, phonological, and morphological properties.

Before describing Amharic phonetics, phonology, syntax and writing system it is appropriate to view the basic terms [11].

Phonetics is the study of speech sounds used in languages of the world. It is concerned with sounds of languages, how these sounds are articulated and how the hearer perceives them. Phonetics is related to the science of acoustics in that it uses much of the techniques used by acoustics in the analysis of sound [23].

Phonology is the study of the sound patterns of a language. It describes the systematic way in which sounds are differently realized in different contexts, and how this system of sounds is related to the rest of the grammar. Phonology is concerned with how sounds are organized in a language. It `endeavors to explain what these phonological processes are in terms of formal rules.

Morphology is the study of word formation and structure. It studies how words are put together from their smaller parts and the rules governing this process. The elements that are combined to form words are called morphemes. A morpheme is the smallest meaningful unit you can have in a language.

Syntax is the study of sentence structure. It attempts to describe what is grammatical in a particular language in terms of rules. Syntactic knowledge of a language is given to an ASRS as its language model.

3.4 Basics of Amharic Phonetics

Articulatory phonetics shows that characteristics of sound are determined by the position of the various articulators in the vocal tract and the state of the vocal cords [23]. Three aspects are used to show the phonetics of Amharic language: Voicing, Manner and Place of articulation. Linguists decompose a spoken language into elements of linguistically distinct sounds called phonemes. According to [23] Phonemes are determined and classified according to their corresponding articulatory configurations. According to the voicing aspect sounds classified as voiced and unvoiced. In the articulation of manner sounds are classified in the classes: Stops, Fricatives, and Approximants. Place of articulations includes Labial, Dental, Palatal, Velar and Glottal. Sounds can also be classified as vowels and consonants. Vowel and consonant sounds are produced in fundamentally different ways.

3.4.1 Consonant phonemes

Consonants, as opposed to vowels, are characterized by significant constriction or obstruction in the pharyngeal and/or oral cavities, that some consonants are voiced; others are not. A set of 34 phones, seven vowels/**አናባቢ/** and 27 consonants/**ተነባቢ/**and 7other repeated consonants, makes up the complete inventory of sounds for the Amharic language [23]. Amharic consonants according to manner of articulation are generally classified as stops/**ዝግ/**, fricatives/**ተሸሉክሳ.ከ.** /- formed when the stricture is very narrow (but without total closure) so that when air flows out, a hissing noise is made, and lateral- is made when air flows out of the sides of the mouth, nasals/

የአፍንጫ/- air to flow out of the nose, liquids/ **የምላስ/**, and semi-vowels/ **ግማሽአናባቢ/**. Figure 3.2 shows the phonetic representation of the consonants of Amharic as to their manner of articulation, voicing, and place of articulation are of voiceless //**የማይነገር**/, voiced/ **ነዛሪ/**, and glottal/ **የጉሮሮ/**.

Amharic consonants are generally classified based on their voicing, manner, and place of articulation. In articulatory phonetics, the place of articulation (also point of articulation) of a consonant is the point of contact where an obstruction occurs in the vocal tract between an articulatory gesture, an active articulator (typically some part of the tongue), and a passive location (typically some part of the roof of the mouth). Figure 3.1 shows the phonetic

representation of the consonants of Amharic as to their manner of articulation, voicing, and place of articulation.

Manner of Articulation	Voicing	Place of Articulation											
		Labials		Alveolar		Palatals		Velars		Labio-Velar		Glottals	
Stops	Voiceless	p	ፕ	t	ት			k	ክ	kx	ኩ	ax	ዕ
	Voiced	b	ብ	d	ድ			g	ግ	gx	ጉ		
	Glottalized	px	ጵ	tx	ጥ			q	ቅ	qx	ቋ		
Fricatives	Voiceless	f	ፍ	s	ስ	sx	ሽ					h	ሀ
	Voiced	v	ቭ	z	ዝ	zx	ዥ						
	Glottalized			xx	ጽ							hx	ኋ
Africatives	Voiceless					c	ች						
	Voiced					j	ጅ						
	Glottalized					cx	ጭ						
Nasals	Voiced	m	ም	n	ን	nx	ኝ						
Liquids	Voiced			l	ል								
	Voiced			r	ር								
Glides		w	ው			y	ይ						

Table 3.1 Phonetic representation of Amharic consonants adopted from [24].

3.4.2 Vowel phonemes

Vowels are always voiced sounds and they are produced with the vocal cords in vibration. Most languages have five vowels/a, e, i, o, u/, but in case of Amharic, there are seven vowels see Figure 3.3) below. In addition to the five vowels which is common for different languages, Amharic has two central vowels, /e/ and /ix/, the latter with a mainly epenthetic function, used for epenthesis vowel insertion/. It is impossible to analyze the effects of vowels when we are speaking, rather we can observe clearly the effects of those when we write, due to that listener sees only written form, rather than spoken form. The effects are clearly analyzed and understood during acoustic formation of sounds as in CV-syllable assimilation.

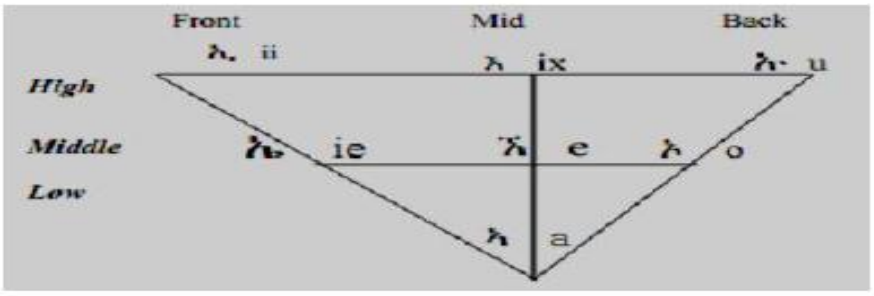

Figure3. 2 Phonetic representations of Amharic Vowels adopted from [24].

3.5 Syllable structure of Amharic

Syllables are very important unit of language, without the complete knowledge of syllables there is no possibility to apply linguistic in speech applications [25]. Syllables are the combination of consonants and vowels. Amharic words use consonantal roots with vowel variation expressing difference in interpretation. In modern written Amharic language, each syllable pattern comes in seven different orders, reflecting the seven vowel sounds. The first order is the basic form; the other orders are called derived from which is formed from more or less regular modification using those seven vowels to drive others. Amharic has its own non-Latin based syllabic script called "Fidel" or "Abugida". The Ethiopian script is not strictly speaking an alphabet, but what is called a syllabary. This means that each letter or symbol usually represents a whole syllable. There are thirty-three basic shapes, represent the consonants followed by the seven vowels (a, e, i, o, u, e and ix).The basic shapes are altered in various ways to indicate a different vowel following the base consonant. The Amharic syllabary is usually presented as a grid with the vowels in the horizontal axis/columns and the consonants in the vertical axis/rows.

3.6 Syllabification

A syllable is a basic unit of word studied on both the phonetic and phonological levels of analysis. It is typically composed of more than one phoneme [25]. No matter how easy it can be for people and even for children to count the number of syllables in a sequence in their native language, still there are no universally agreed upon phonetic definitions of what a syllable is. It is phonologically believed that syllable is a complex unit made up of nuclear and marginal elements. There is general agreement that a syllable consists of a nucleus that is almost always a vowel, together with zero or more preceding consonants (the onset) and zero or more following consonants (the coda) but determining exactly which consonants of a multisyllabic word belong to which syllable is problematic. Nuclear elements are the vowels or syllabic segments, and marginal elements are the consonants or non-syllabic segments. Standard dictionaries provide syllabification that is influenced by the morphological structure of words; it is common in such dictionaries to split prefixes and suffixes from stems [25].

Amharic claimed as a syllabic language in which every grapheme (character) represents (Consonant-Vowel Assimilation). However, while reading a text in Amharic, all the syllables are not uttered as expected and hence the text syllables are not the CV sequence seen in the text. This limits performance of many speech systems and other NLP applications. A great number of diverse algorithms have been proposed for syllabification in different languages and many researches has been done on other languages. In many languages, the pronunciation of phonemes is a function of their location in the syllable relative to the syllable boundaries. Location in the syllable also has a strong effect on the duration of the phone and on the temporal alignment of the fundamental frequency contour with the segmental chain, and is therefore a crucial piece of information for segmental duration and intonation models [25].

The complexity of syllable onset and coda structure poses serious problems for a syllabification algorithm because despite restrictions as to which consonants, or classes of consonants, may occur in any given position within the onset or coda of a syllable-ambiguous and multiple alternative syllable boundary locations are usually observed in polysyllabic words, notably in. It has been identified that there are six legal syllable structures (templates) in Amharic, namely V, VC, VCC, CV, CVC and CVCC for words [24], which is the most convenient template of syllable structure for Amharic

CHAPTER FOUR

EXPERIMENTATION AND ANALYSIS

With this study, in order to increase the performance for Amharic spontaneous speech recognition system different experiments has been carried out. We follow various mechanisms like the cross-word tri-phone, the use of trigram language model, changing the sub-word unit to CV-syllable, and parameter tuning techniques. The detail of the experimentation is presented and discussed below.

4.1 System Architecture.

According to [21], there are four main steps in developing a speech recognizer using HTK. Namely: data preparation, training, recognition and analysis. The four major steps to develop speech recognizer are presented below in Figure 4.1 with training and testing phases [21].

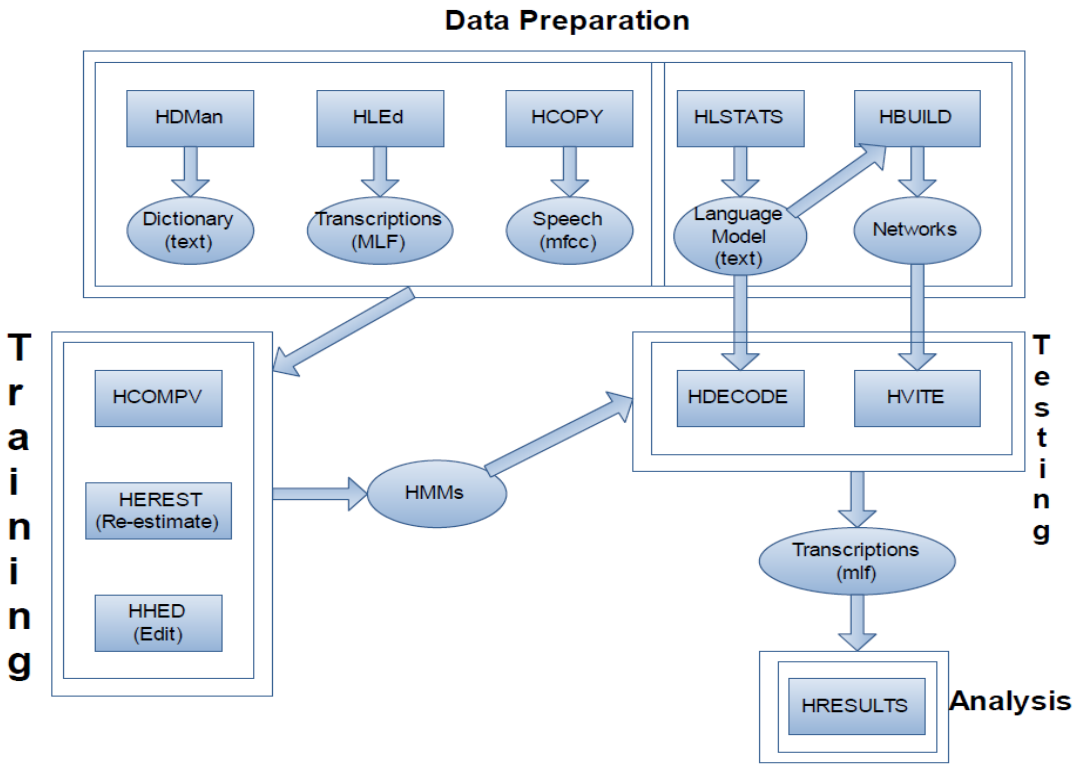

Figure 4. 1Architecture of the system (Adopted from [21]).

The system architecture is adopted from [21] the same as other who done in this area what make it different is techniques, tool, and method we follow is different. We use HDEcode to construct the cross-word tri-phones and we build the tri-gram language model using SRILIM. With regarding to the methods we use a CV-syllable unit as a unit of speech recognition others researcher who done in this area uses phone as a unit of recognition.

4.2 Data preparation

Data preparation is the first task that should be addressed in speech recognition development process. The first stage of any recognizer development project is data preparation. Speech data is made between two and more speakers in the form of conversation. In the case of the training data, the prompt scripts will be used in conjunction with a pronunciation dictionary to provide the initial phone level, and syllable level transcriptions needed to start the HMM training process. We use the spontaneous speech corpus prepared by [10] for both training and testing.

For training we have done modification like replacing irregular symbols with other characters. Such as in the spontaneous speech corpus of [10] there is a symbol ~ to represent hesitation we replace this symbol by HES the same for other non-speech events also. To increase the corpus size we add the spontaneous speech corpus of [9] to see its effect on the performance of the recognizer. Before increasing the corpus size by adding the spontaneous speech corpus of [9] since the corpus is prepared using the SPHNIX tool symbols like ++NOISE++ , ++FP++ are replaced by suitable characters convenient for HTK tools and the transcription file of [9] is Unicode, so we transliterate to its equivalent ASCII. The speech data (wav) file of [9] is listed with another file names for the speaker which is different from the HTK format we arrange the file format to make it suitable for HTK tool. In Data preparation stages we prepare the pronunciation dictionary and extract the feature file by converting the .wav format to .mfc format using the HCopy to make it ready for training.

4.2.1 Pronunciation Dictionary

Two type of pronunciation dictionaries are prepared for training and testing the performance of the recognizer. The pronunciation dictionary based on the sub-word unit phone, and the pronunciation dictionary based on the sub-word unit syllable. In both cases the dictionary editor HDman is used to prepare the pronunciation dictionary. To create a pronunciation dictionary in HTK we need to have prompts file. This is the file with the list of utterances (sentences). The

first step in building a dictionary is to create a sorted list of the required words using the Perl script prompts2wlist. The HDMan command is used to go through the word list (wordlist) file and look up the pronunciation for each word from a separate pronunciation files we have prepared for both speech and non-speech events, and output the result in a Pronunciation Dictionary.

Since we use spontaneous speech corpus of [10] only four non-speech events are included in the corpus like filled pause, hesitation, breath, and OTH are modeled in two ways. For the phone based pronunciation dictionary they are modeled as phone and for syllable based pronunciation dictionary they modeled as syllable.

HDMan -m -w wordlist -n monophones1 -l dlogdictspeech_dict.lexnonspeech_dict.lex

The above command creates a new dictionary called *dict* by searching the source dictionaries,

speech_dict.lex and *nonspeech_dict.lex* developed using python code we have developed for this purpose to find pronunciations for each word in *wordlist.* See Appendix A for sample syllable based pronunciation dictionary.

The general format of each dictionary entry is: WORD [outsym] p1 p2 p3 …This means that the word WORD is pronounced as the sequence of phones p1 p2 p3 … the string in square brackets ([outsym]) specifies the string to output when that word is recognized. If it is omitted then the word itself is output. If it is included but empty, then nothing is output.

Phone-based pronunciation dictionary

[BR]	BR sp	
CEbITEwI	[CEbITEwI]	C E b I T E w I sp
CEle	[CEle]	C E l e sp
CEmErIkI	[CEmErIkI]	C E m E r I k I sp
CEmErIkuNI	[CEmErIkuNI]	C E m E r I k u N I sp
CEmIrEhI	[CEmIrEhI]	C E m I r E h I sp

```
CEbITEwI       [CEbITEwI]      CE bI TE wI sp
CElEmEmIbINI   [CElEmEmIbINI]  CE lE mE mI bI NI sp
CEle           [CEle]          CE le sp
CEmErIkI       [CEmErIkI]      CE mE rI kI sp
CEmErIkuNI     [CEmErIkuNI]    CE mE rI ku NI sp
CEmIrEhI       [CEmIrEhI]      CE mI rE hI sp
```

CV syllable based pronunciation dictionary

4.2.2 Transcription

Word Level Transcriptions: - To train a set of HMMs, every file of training data must have an associated phone level transcription. Since there is no hand labeled data to bootstrap a set of models, a flat-start scheme have been used instead. To do this, two sets of phone transcriptions are needed. The set used initially have no short-pause (sp) models between words. Then once reasonable phone models have been generated, an SP model is inserted between words to take care of any pauses introduced by the speaker. Since HTK toolkit cannot process prompts file directly, for both sets of phone transcription an orthographic transcription in HTK Label format is needed. To do this we have two options; option one, we can create a separate 'label' file for each line of our prompts file using a text editor. The second option, we can create a Master Label File (MLF) using scripting language. MLF file is a single file that contains a label entry for each line in prompts file. Thus, MLFs allow large sets of files to be stored in a single file, they allow a single transcription to be shared by many logical label files and they allow arbitrary file redirection. This is the easiest approach, and the one we will use for this study. For this purpose we have used the Perl script prompts2mlf to generate the mlf file from our prompts file.

Syllable Level Transcriptions

Next you need to execute the HLEd command to expand the Word Level Transcriptions to syllable Level Transcriptions - i.e. replace each word with its syllable, and put the result in a new syllable Level Master Label file. Thisis done by reviewing each word in the MLF file, and looking up the syllable that make up that word in the dict file you created earlier, and outputting

the result in a file called syllable0.mlf (which will not have short pauses ("sp"s) after each word phone group)

Phone Level Transcriptions

In this step the HLEd command is executed to expand the Word Level Transcriptions to Phone Level Transcriptions - i.e. replace each word with its phones, and put the result in a new Phone Level Master Label file. This is done by reviewing each word in the MLF file, and looking up the phones that make up that word in the dict file you created earlier, and output the result in a file called phones0.mlf (which will not have short pauses ("sp"s) after each word phone group).After extracting the feature,i.e. Making it suitable for training we can go through training process to build the acoustic model.

4.2.3 Feature extraction

The final stage of data preparation is to parameterize the raw speech waveforms into sequences of feature vectors. HTK is not efficient in processing *wav* files as it is with its internal format. Therefore, we need to convert our audio *wav* files to mfc file format we done this using HCOPY is a general-purpose tool for copying and manipulating speech files. The general form of invocation isHCopysrctgtHTK support both FFT-based and LPC-based analysis. Here we have used Mel Frequency Cepstral Coefficients (MFCCs), which are derived from FFT-based log spectra, and one of the more common techniques of studying a speech signal via the power spectrum.To get the final extracted feature vector first we have to extract at the word level. After extraction at the word level it's possible to extract to sub-word unit (phone level or syllable level).

4.3 Training the Model

Four phase of experimentation is conducted. In the first three phase to train the acoustic model we used a corpus of 3hour and 10 minutes spontaneous speech which is a total of 2007 (two thousand seven) utterance. In the fourth phase we used spontaneous speech corpus of 4hour and 30minutes which is a total of 3556 utterance. In both cases throughout the experimentation the pronunciation dictionary is canonical dictionary; which has a total of 9236 (four thousand two hundred and three) unique words for the phone based, and 12306(Twelve thousand three hundred six) for syllable based.

The total phone coverage used in this study is (forty two) phones, and 238 syllable unit including the four non-speech events.

4.3.1 Creating Mono-phone HMMs

Prototype Definition: - The first step in HMM training is to define a prototype model. The parameters of this model are not important; its purpose is to define the model topology. Our recognition system is tested using phone-based system, and syllable-based. We have done training with topology 5-state (3 emitting states) left to right with no skips and with skips for the phone based and,7-state (5 emitting states) left-right with no skip for the syllable based in both topologies the starting and ending states are non-emitting states. To define the topology we have created the file proto, which is in appendix H.

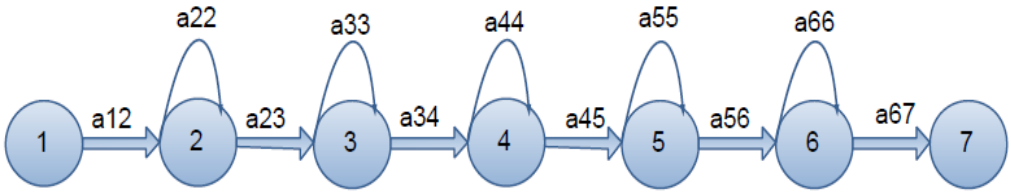

Figure4. 2HMM model with 5 emitting state adapted from [1].

4.3.2 Re-estimating monopones and Syllable

The mono-phones, and the flat syllable created are re-estimated using the embedded re-estimation tool HERest invoked as follows: -

```
HERest -C config -I phones0.mlf -t 250.0 150.0 1000.0 -S train.scp
-H hmm0/macros -H hmm0/hmmdefs -M hmm1 monophones0
```

```
HERest -C config -I phones0.mlf -t 250.0 150.0 1000.0 -S
train.scp -H hmm0/macros -H hmm0/hmmdefs -M hmm1
syllable0
```

The above command loads all the models both *hmmdefs*and *macros* which are listed in the model list. Mono-phones, and syllable used here are excluding the short pause (sp) model. These are then re-estimated using the data listed in *train.scp*and the new model set is stored.

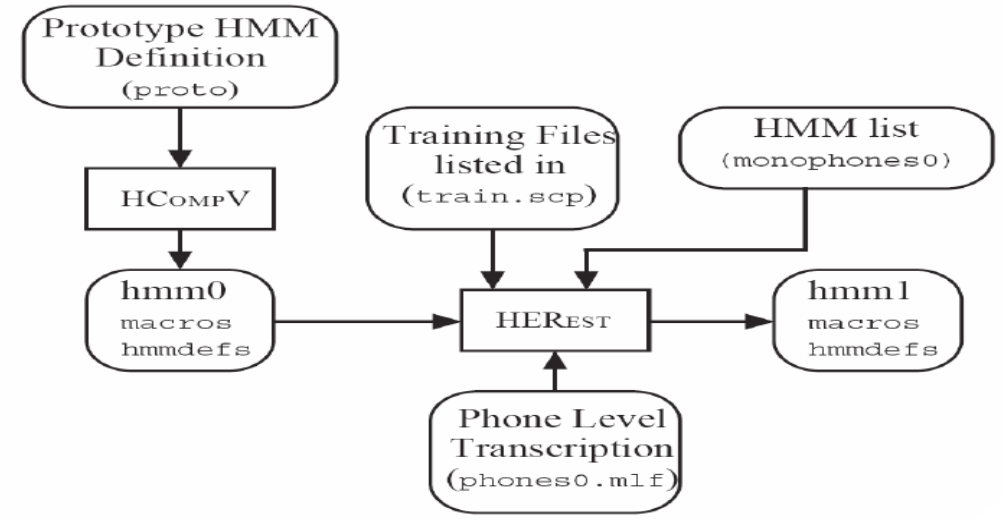

Figure4. 3Creating flat-start mono-phones adapted from [1].

Fixing the Silence

With the previous steps we have generated a 3 state left-to-right HMM for each phone and also for the silence model *sil*. The next step is to add extra transitions from states 2 to 4 and from states 4 to 2 in the silence model. The idea here is to make the model more robust by allowing individual states to absorb the various impulsive noises in the training data. The backward skip allows this to happen without committing the model to transit to the following word. Also, at this point, we have created 1 state short pause *sp*model. This sp has its emitting state tied to the center state of the silence model. The required topology of the two silence models is shown in Figure 4.4

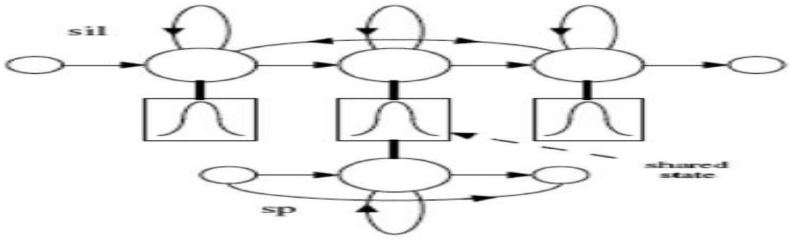

Figure4. 4 Silence Models Adapted from [1].

These silence models has been created by running the HMM editor HHEd to add the extra transitions required and tie the sp state to the center sil state. HHEd works in a similar way to

HLEd. It applies a set of commands in a script to modify a set of HMMs. In this case, it is executed as follows, where sil.hed contains the following commands

```
HHEd -H hmm4/macros -H hmm4/hmmdefs -M hmm5 sil.hed monophones1
```

```
AT 4 2 0.2 {sil.transP}
AT 1 3 0.3 {sp.transP}
TI silst  {sil.state[3],sp.state[2]}
```

The AT commands add transitions to the given transition matrices and the final TI command create a tied-state called *silst*. The parameters of this tied-state are stored file and within each silence model, the original state parameters are replaced by the name of this macro. Macros are the mechanism by which HTK implements parameter sharing. The new list of mono-phones contains sp model is used in the above HHEd command. Finally, HERest are applied using the phone transcriptions with sp models between words.

4.3.3 Refinements and Optimization

A. Tied-State Tri-phones

As stated by [1] the first stage of model refinement is usually to convert a set of initialized and trained context-independent mono-phone HMMs to a set of context dependent models. But before building a set of context-dependent models, it is necessary to decide whether or not cross-word tri-phones are to be used. If they are, then word boundaries in the training data can be ignored and all mono-phone labels can be converted to tri-phones. If word internal tri-phones are used, then word boundaries in the training transcriptions must be marked. We have built both cross-word and word internal tri-phones for this study.

```
AT 4 2 0.2 {sil.transP}
AT 1 3 0.3 {sp.transP}
TI silst {sil.state[3],sp.state[2]}
```

Since we have prepared a set of mono-phone HMMs with the previous steps, now we can use them to create context-dependent tri-phone HMMs. We did this in two steps. Firstly, the mono-phone transcriptions are converted to tri-phone transcriptions and a set of tri-phone models are

Created and re-estimated. Secondly, similar acoustic states of these tri-phones are tied. Tying means nothing but it is the method of making one or more HMMs share the same set of parameters. Then the set of context-dependent models re-estimated using HERest tool.

Making Tri-phones from Mono-phones

Context-dependent tri-phones can be created from mono-phones. The tri-phone transcription have been created first using HLEd tool which enables us to generate a list of all the tri-phones for which there is at least one example in the training data.

HLEd-n triphones1 -l '' –I wintri.mlfmktri.ledwords.mlf*

Using the above command we have created the tri-phone transcriptions in *wintri.mlf*file using mono-phone transcriptions *words.mlf*file. At the same time, a list of tri-phones is written to the file *triphones1*.

The edit script *mktri.led*used above contains the commands:

```
WB sp
WB sil
TC
```

The two WB command defines *sp*and *sil*as word boundary symbols. These then blocks the addition of context in the TI command, which converts all phones (except word boundary symbols) to tri-phones. For example, Sil n E w I sp … becomes siln+En-E+wE-w+I w-I sp … This tri-phone transcription is word internal. Some bi-phones may be generated as contexts at word boundaries since sometimes they include only two phones.

HHEd -B -H hmm9/macros -H hmm9/hmmdefs -M hmm10 mktri.hed monophones1

Where the edit script *mktri.hed*contains a clone command CL followed by TI commands to tie all of the transition matrices in each triphone set, that is: CL triphones1

```
CL triphones1
TI T_a {(*-a+*,a+*,*-a).transP}
TI T_b {(*-b+*,b+*,*-b).transP}
TI T_C {(*-C+*,C+*,*-C).transP}
TI T_c {(*-c+*,c+*,*-c).transP}
```

We have generated the file mktri.hed using the Perl script *maketri.hed* included in the HTK Tutorial directory. The clone command CL takes as its argument the name of the file containing the list of tri-phones (and bi-phones) generated above. For each model of the form *a-b+c*in this list, it looks for the monophone b and makes a copy of it. Due to the latter use of transition

matrix *transP*which is regarded as a sub-component of each HMM, Each TI command takes as its argument the name of a macro and a list of HMM components. The lists of items within brackets are patterns designed to match the set of tri-phones, right bi-phones and left bi-phones for each phone.

Making Tied State Tri-phone

After a set of tri-phone HMMs with all tri-phones in a phone set sharing the same transition matrix prepared now we can tie them.

Tying states within tri-phone sets helps to share data and thus be able to make robust parameter estimates. HTK tool HHEd provides two mechanisms which allow states to be clustered and then each cluster tied. The first is data-driven and uses a similarity measure between states. The second uses decision trees and is based on asking questions about the left and right contexts of each tri-phone. The decision tree attempts to find those contexts which make the largest difference to the acoustics and which should therefore distinguish clusters. Decision tree state tying is performed by running HHEd:

HHEd -B -H hmm12/macros -H hmm12/hmmdefs -M hmm13 tree.hed triphones1

*Mkclscript*script which is found in the RM Demo is used for creating the TB commands (decision tree clustering of states) which is one part of *tree.hed.*

Firstly, the RO command is used to set the outlier threshold to 100.0 and load the statistics file generated at the end of the previous step. The outlier threshold determines the minimum occupancy of any cluster and prevents a single outlier state forming a singleton cluster just because; it is acoustically very different to all the other states. The TR command sets the trace level to zero in preparation for loading in the questions. Each QS command loads a single question and each question is defined by a set of contexts. For example, one of the QS command defines a question called *Nasal'* which is true if the Right context is either of the nasals n, N,or m. In *tree.hed*file using QS command the questions referring to both the right and left contexts of a phone are included. The full set of questions loaded using the QS command would include every possible context which can influence the acoustic realization of a phone, and can include any linguistic or phonetic classification which may be relevant. There is no harm in creating extra unnecessary questions, because those which are determined to be irrelevant to the data will be ignored. For this study we have taken on and used the questions (QS) with some

modifications from Adugna [10]. The edit script *tree.hed*contains the instructions regarding which contexts to examine for possible clustering, see appendix F for more.

```
RO 100.0 stats
TR 0
QS "R_NonBoundary" { "*+*" }
QS "R_Silence" { "*+sil" }
QS "R_Stop" {
"*+b","*+d","*+g","*+k","*+P","*+p","*+q","*+t","*+T","*+ua","*+H" }
QS "R_Nasal" { "*+m","*+n","*+N" }
TR 2
TB 600 "ST_BR_2_" {("BR","*-BR+*","BR+*","*-BR").state[2]}
TB 600 "ST_C_2_" {("C","*-C+*","C+*","*-C").state[2]}
TB 600 "ST_E_2_" {("E","*-E+*","E+*","*-E").state[2]}
TB 600 "ST_b_2_" {("b","*-b+*","b+*","*-b").state[2]}
…
TR 1
AU "fulllist"
CO "tiedlist"
ST "trees"
```

The set of tri-phones used so far only includes those needed to cover the training data. The AU command takes as its argument a new list of tri-phones expanded to include all those needed for recognition. This list can be generated, for example, by using HDMan on the entire dictionary (not just the training dictionary), converting it to tri-phones using the command TC and outputting a list of the distinct tri-phones to a file using the option –n.

HDMan -b sp -n fulllist -g global.ded -l flog beep-tri beep

The *-b sp* option specifies that the *sp* phone is used as a word boundary and it is excluded from tri-phones. The effect of the AU command is to use the decision trees to synthesize all of the new previously unseen tri-phones in the new list. Once all state-tying has been completed and new models synthesized, some models may share exactly the same states and transition matrices and they became identical. The CO command is used to compact the model set by finding all identical models and tying them together, producing a new list of models called *tiedlist.*

One of the advantages of using decision tree clustering is that it allows previously unseen tri-phones to be synthesized. To do this, the trees must be saved and this is done by the ST command. Finally, the models are re-estimated using HERest.

Making Cross-word tri-phones

In order to make cross word tri-phones first a set of cross word labels and cross-word tri-phone model is needed. Both cross-word tri-phone models list and label files are generated using HLEd tool and the mono-phone label file we have created.

*HLEd -l * -n xwrd.list –I xwrd.mlfxwrd.hled phones0.mlf*

The command creates *xwrd.list*file which is list of cross-word tri-phones and *xwrd.mlf*cross-word tri-phones label file using editing file *xwrd.hled*. *xwrd.hled*contains the following commands:

```
NB sp
TC
IT
RE silsil
RE spsp
```

Next an initial set of cross-word triphone models are created by cloning the mono phone models using HHEd. We have used a HHEd edit file *clone.hed*containing the following commands

```
MM "trP_" { *.transP }
CL "xwrd.list"
```

HHEd -B -T 1 -H hmm0/models -w hmm1\MODELS clone.hed monophones1

Once cross-word tri-phone models have been created and tied, then new cross-word tri-phone set re-estimated using HERest. This is done as previously done except that the mono-phone model list is replaced by a cross-word tri-phone list and the cross-word tri-phone transcriptions are used in place of the mono-phone transcriptions.

B. Increasing Gaussian mixture

In the previous stages the construction of mono-phones, context dependent tri-phones and cross-word tri-phones are done with single Gaussian models. But since the increment of Gaussian mixture has a significant effect on the performance of recognizer it is better to increase the Gaussian mixture of the models we have constructed in previous tasks. In HTK, the conversion

from single Gaussian HMMs to multiple mixture component HMMs is usually one of the tasks in a system refinement. The mechanism provided to do this is the HHEd tool *MU* command which helps to increase the number of components in a mixture by a process called mixture splitting.

This approach to building a multiple mixture component system is extremely flexible since it allows the number of mixture components to be repeatedly increased until the desired level of performance is achieved. The MU command has the following form:

MU n itemList

Where *n* is the new number of mixture components required and *item List* defines the actual mixture distributions to modify. For example, to increases the number of mixture components in the output distribution for state 2 to 4 of all models to 2. We have used the following MU command:

MU 2 {.state[2-4].mix}*

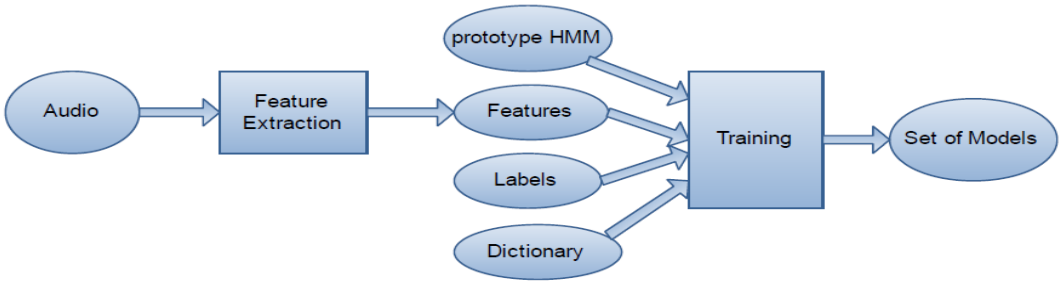

Figure 4.5 Summary of one time training process adopted from [1].

Language Model

In order to develop the language model two types of tools are used; Hlstats with Hbuild to build the bi-gram language model, and the SRILIM toolkit used to build tri-gram language model. Both type of the language model which mentioned above is used for recognition. The bigram language model is taken from [10]. And the trigram language model is built using the text corpus prepared by Solomon [28]. We use external decoder HDecode to use the tri-gram language model. The decoder distributed with HTK, HVITE, is only suitable for small and medium vocabulary system and systems using bigrams. For larger vocabulary systems, or those requiring trigram language models to be used directly in the search, HDECODE is available as an extension to HTK. HDECODE has been specifically written for large vocabulary speech recognition using cross-word tri-phone models.

4.4 Recognizer testing and evaluation

The heart of automatic speech recognition is the search for the most likely word sequence given the observed features extracted from the speech signal. This is commonly referred to as decoding or recognizing the speech signal. When decoding speech, we begin by constructing a search graph which contains every word in the recognition vocabulary. Each word is then replaced by the HMMs that correspond to the sequence of sound units which make up the word. As a result, the search graph is a large HMM, and recognition is performed using the Viterbi algorithm to align the search graph to the speech features derived from the utterance. Because the Viterbi algorithm is used to find the most likely word sequence, the decoding procedure is said to be done via Viterbi search. We have prepared both test and training data, we have built language and lexical models and we have performed training which comes up with Acoustic model in our previous activities. Now we can say that the recognizer is built and we can perform recognition and then evaluate the result. Here for recognition HTK tools HVite for word internal tri-phone and HDecode for cross word tri-phone are used.

4.5 Comparisons of Result and Discussion

The Amharic spontaneous speech recognizer we have developed in this study has been tested using test data consisting of 104 utterances which is composed of around 820 unique words. The sentences (utterances) are from 14 speakers which 5 of them are females and 9 of them are males. Among these 14 speakers, 10 of them were speakers which were involved in training but 4 of them are not. The language model we have used for word probability was bigram, and tri-gram. In order to improve the recognizer performance we increased the number of Gaussian mixture then trained repeatedly and come up with acoustic models with different mixtures. Therefore all the results we have reported are the results found at twelfth mixture. The results are given below in tables and followed by discussion. The experimentation is conducted in four major phases each phases were tested using speaker independent and mixed test set. Finally, the result is reported according to the experimentation.

We use two type of test set already prepared by Adugna [10] which is collected from a total of 14 speakers. The two type of test set we use throughout the experimentations are mixed test set and speaker independent test set. Their characteristics are described as follows:- Mixed test set :- the test set is prepared by Adugna[10] these test set includes speaker involved in training also participated during testing of the recognizer performance , in such type of test set the

recognition result is higher than speaker independent test set because, in this test set 10 speaker involved in training is included in testing so it increase the chance of recognition rate of the unknown utterance . Speaker dependent systems can result in word error rates 2–3 times lower than Speaker independent systems (given the same amount of training data) [5].

Speaker independent test set: - The other test set is also prepared by Adugna[10] which excluded the Speaker participating in training from testing , in such type of system recognition result is lower than Speaker dependant test set.

All the report is generated using the two test set by holding different experimentation like tuning techniques , Cross-word tri-phone , tri-gram language model, CV-syllable as recognition unit and finally, by increasing the corpus size the result is reported accordingly.

4.5.1. Using parameter tuning techniques

The first experiment we conduct is using the tuning techniques to know the effect of the parameter values on recognizer performance. The experimentation is conducted by changing the parametric values for the search space (pruning), the insertion penalty -p, and for the grammar scale factor (language weight) -s. By using the same training, test data set, and using the same language model which is prepared by Adugna [10] only by changing the parametric values for the grammar scale factor-s, insertion penality –i, and search space-t the performance of the recognizer is evaluated. The list of the parameter which yields good recognition result listed in Table 4.1 below.

Speaker code	Grammar scale factor(-s)	Insertion penalty (-i)	Word %	Accuracy%
tst 001	15.0	0.1	56.43	40.65
tst 002	15.0	0.1	67.68	58.27
tst 003	18.0	0.2	67.45	43.79
Tst 004	19.0	0.2	58.03	42.28
Tst 006	17.0	0.1	57.73	41.27
*Tsts 007	21.0	0.2	38	20.65
Tst 008	19.0	0.1	57.14	44.16
tst 009	18.0	0.2	64.57	49.61
Tst 010	15.0	0.1	54.79	47.95
Tst 011	19.0	0.1	59.76	41.18
*Tst 012	16.0	0.2	42.68	26.65
*Tst 013	18.0	0.1	37.45	29.19
*Tst 014	15.0	0.1	48.03	38.88
Average	15.0	0.1	57.47	41.06

Table4.1 Result of Mixed test set using tuning techniques.

The result shows that the tuning techniques using the mixed test set from the participating 14 Speaker the one which is indicated by the * symbol is to show speaker independent test set ten speaker which involved in training is also participated in testing.

The option -p and -s set the word insertion penalty and grammar scale factor, respectively. The word insertion penalty is a fixed value added to each token when it transits from the end of one word to the start of the next. 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. These parameters can have a significant effect on recognition performance. Among the four parameters especially the grammar scale factor highly co-related with recognition performance. We manually change the parameter values from 15.0 to 25.0 since

we get these ranges are suitable range for the grammar scale factor the amount by which the language model is scaled from those parameter values. From those values when we set 15.0 for –s we get good recognition result.

We get good recognition result in tst002 when -s is set to 15.0, we get good recognition result for tst009 when -s is set to 18.0. Changing the pruning threshold value for the search space -t did not have significant impact on the performance of the recognizer which normally 250.0 if re-estimation fail on any particular file the threshold is increased by 150.0 and the file is re-processed up to 9000 we increment the search space by 500. This is repeated until either the file is successfully processed or the pruning limit of 1000.0 is exceeded.

The insertion penalty –p (between 0.1 and 0.2), and -s for the grammar scale factor has significant effect on the performance of the recognizer for the mixed set by changing the parameter we get percentage of correctly recognized word 57.47% and the accuracy is 41.06% which shows we 2.47 word % and 2.06 % accuracy performance improvement from the result of mixed test set of Adugna [10].

We done other experimentation using other test set which is speaker independent test set using tuning techniques. Table 4.2 below depicts experimental result of speaker independent.

Speaker code	Grammar scale factor(-s)	Insertion penalty(-i)	Search space(-t)	Word %	Accuracy%
*Tst 007	21.0	0.2	250.0 150.0 4000.0	38	20.65
*Tst012	16.0	0.2	250.0 150.0 800.0	42.68	26.65
*Tst013	18.0	0.1	250.0 150.0 8500.0	37.45	29.19
*Tst014	15.0	0.1	250.0 150.0 9000.0	48.03	38.88
Average	15.0	0.1	250.0 150.0 7500.0	41.54	28.84

Table4.2 Result of speaker independent test set using tuning techniques.

We follow the same procedure like we did for the mixed test set what make it different is that we use speaker independent test set to check the effect of the parameter in the recognizer performance.

For speaker independent test set we get correctly recognized word 41.54, and 28.84% accuracy by changing the parametric we come up with 2.54 word % and 3.24% accuracy improvement from the result reported by Adugna [10].

4.5.2. Exploring the effect of cross-word tri-phone and tri-gram language model

This experiment is conducted in order to explore the effect of cross-word tri-phone and tri-gram language model on the performance of the recognizer. First we construct tri-gram language model using the SRILIM toolkit since HVite are support bi-gram language model, and word internal tri-phone we use external decoder HDecode to see the effect of cross-word tri-phone using trigram language model on the performance of the recognize. Since cross-word doesn't support special characters like -INT-, ¬, and ~ such kinds of symbols have been replaced by other character such as ~ is replaced by HES to represent hesitation then it is possible to form tri-gram language model, the Hdecode decoder is used to test speaker independent test set then it shows improvement from the previous word-internal tri-phone using the Hvite recognizer. In order to make cross word tri-phones first a set of cross word labels and cross-word tri-phone model is needed. Both cross-word tri-phone models list and label files are generated using HLEd tool and the mono-phone label file we have created.

HLEd -l * -n xwrd.list –I xwrd.mlfxwrd.hled phones0.mlf

The command creates xwrd.list file which is list of cross-word tri-phones and xwrd.mlf Cross-word tri-phones label file using editing file xwrd.hled. xwrd.hled .The step to create cross-word tri-phone is attached in the appendix G.

In this experiment we explore the effect of cross-word and making tri-gram language model separately. In the first case we report the result only by making cross-word tri-phone these enable us which techniques clearly increase the performance of the recognizer. Before moving to cross-word tri-phone result by tracing the parameter value we select best parment for the HDEcode decoder. Then we are done the cross-word tri-phone experiment for the HDEcode decoder the grammar scales factor and the pruning threshold that led to the best performance were selected as follows in Table 4.3

No.	-t	-p	-s	Word%	Accuracy %
1	500.0 8000.0	0.1	18.0	29.75	24.79
2	300.0 8000.0	0.1	18.0	40.17	28.00
3	250.0 10000.0	0.2	18.0	40.19	27.53
4	300.0 1000.0	0.1	18.0	38.17	26.32
5	300.0 1000.0	0.2	15.0	30.25	22.69
6	370.0 8000.0	0.1	18.0	40.08	28.42
7	370.0 10000.0	0.1	18.0	46.08	32.42
8	270.0 10000.0	0.1	16.0	28.95	21.05

Table4.3Parameter list of cross-word tri-phone.

The table shows that good recognition result achieved in no 7 when the search space-t set between 370.0 10000.0, insertion penality 0.1, and the grammar scale factor is set to 18.0, and then we achieved 46.08 % correctly recognized word, and 32.42% accuracy.

Then we explore the effect of increasing the n-gram model from bi-gram to tri-gram language model. To do that we adjust the configure variable for the Hvite in order to use the cross-word tri-phone by changing the variable FORCEcxtecp to false and ALLowxwrdexp to true. Then we enable to work on the cross-word tri-phone.

Tri Phone used	Language model	Correct recognized word %	Accuracy %
Cross -word tri-phone	Bi-gram	41.25	27.6
Cross -word tri-phone	Tri-gram	46.08	32.42

Table 4.4Comparison between bi-gram and tri-gram language model for speaker independent test set.

Before making comparison we have to be sure about using the same tri-phone model which is cross-word tri-phone in both cases know it is clear that the performance of the recognizer is affected by the language model.

The result show that which language model performs better in recognition performance both for the correctly recognized word as well as the accuracy level. In Table 4.4 we get better recognition result in the tri-gram language model rather than bi-gram, because tri-gram language model is larger in context than bi-gram language model. By using tri-gram language model we come up with 4.75% correctly recognized word and 5% accuracy improvement for speaker independent test set.

A unigram is a window placed over a text, such that we only look at one word at a time. In similar fashion, a bigram shows two words at time. Unlike the two mentioned above, the tri-gram language model is advantageous since it shows three words at a time.

Then we explore the impact of cross-word tri-phone in recognition performance separately. To do that we use the same language model which is bi-gram then we test using different tri-phone model. Using cross-word model and using word-internal model we compare the word-internal tri-phone against the cross-word tri-phone, as shown in table 4.5 below.

Tri Phone used	Language model	Correct recognized word %	Accuracy %
Cross -word tri-phone	Bi-gram	41.25	27.6
Word internal tri-phone	Bi-gram	39.17	23.33

Table 4.5 Comparison between cross word tri-phone and word internal tri-phone.

The comparison is made between word internal tri-phone and cross-word tri-phone using the speaker independent test set. Cross-word tri-phone models co-articulation effect across word boundaries. But in the case of word internal tri-phone word boundaries are not ignored so it is difficult to model co-articulation effect across word boundaries to compare the result we used the same language model which is bi-gram in order to explore the effect of cross-word tri-phone on the performance of the recognizer. We follow different states to build cross-word tri-phone which is attached in appendix G; finally we compare the result with word-internal tri-phone, and we came up with 2% correctly recognized word and 4 % accuracy improvement using cross-word tri-phone.

The same procedure followed to explore the effect of cross-word tri-phone, tri-gram language model what make is different is we use mixed test set in this case. We get the following result (see table 4.5) using the cross-word tri-phone tri-gram language model for the mixed test set.

We get the following result (see table 4.5) using the cross-word tri-phone tri-gram language model for the mixed test set.

Speaker code	Percentage of Word recognized %	Word accuracy percentage%
tst001	61.01	49.13
tst002	56.73	48.86
tst003	58.94	46.81
tst004	56.34	48.72
tst006	55.00	47.36
*tst007	55.41	46.98
tst008	54.97	49.67
tst009	65.97	54.31
tst010	68.92	56.71
tst011	66.48	59.12
tst012	51.43	43.38
tst 013	59.34	44.9
*tst 014	52.1	41.8
Average	58.67	49.13

Table4.6 Result of mixed test set using cross-word tri-phone.

Table 4.6 below present's comparison between cross-word tied state tri-phone, and word-internal tri-phone with the same Gaussian mixture and number of state.

Number of states	NO of Gaussian Mixture	Sub-word unit used	Language model	Word%	Accuracy%
5	12	Cross-word tied state tri-phone	bi-gram	58.67	49.13
5	12	Word-internal tri-phone	bi-gram	55.46	39.86

Table4.7 Comparison between cross-word tri-phone and word internal tri-phone for the mixed test set.

As can be seen from the Table 4.6 cross-word tied state tri-phone using tri-gram performs better than word-internal tri-phone. Cross-word tri-phones (CWTs) are beneficial because they model co articulation effects across word boundaries. The performance analysis shows that, in this study we improve the accuracy of the recognizer by 3.21% word recognition and 9.27% accuracy from the result reported by Adugna[10].

4.5.3. Using CV-syllable

This experiment is conducted since syllable is relatively stable than phones and less context sensitive (co-articulation effect).The experimentation is held using Consonant-Vowel (CV) syllables as recognition units. Even if, there are six syllable structure in Amharic language [24], such as V, VC, VCC, CV, CVC, and CVCC, since the corpus data is not enough to train all syllable structure, and it is a syllabary writing system where each character represents an open CV syllable, i.e., a combination of a consonant followed by a vowel [23], we conducted the experiments only using CV syllables. We prepare a python program to make only CV-syllable; we get 238 CV-syllable including the non-speech event. The non-speech event like FP, OTH,

HES, BR is considered, since they frequently appear in the corpus. They treated as a syllable in this experiment, and the same procedure is followed like we did for creating the monophone. We prepare the pronunciation dictionary for CV-syllable using the HDman, which consists of 9236 unique words represented as CV-syllable in this case the non-speech events are considered as syllable. The first step in hidden Markov model ("HMM") training is defining a prototype model called "proto". The focus here is to create a model structure, the parameters are not important so we model the proto for the CV-syllable as 5 emitting states without skip. Solomon [10] reported that 5 emitting states without skip perform better than 3 emitting states. Then we create the flat start syllable using the Hcompv command. We get a total of 238 CV-syllable including the four non-speech event. The acoustic model is trained using theses 238 syllables. For testing purpose we used 104 sentences uttered by 14 speakers prepared by Adugna[10]. Among the 14 speakers four of them are involved both in training and testing. We get the following result for the CV-syllable based.

Speaker Code	Word%	Accuracy%
tst 001	58.21	47.90
tst002	56.73	46.91
tst003	57.66	41.26
tst004	63.90	40.08
tst006	67.27	48.73
*tst007	57.4	46.2
tst008	58.67	47.56
tst009	52.10	42.58
* tst010	51.06	40.96
*tst011	52.11	40.18
* tst012	55.06	42.48
tst 013	59.7	44.5
tst014	58.75	47.24

Average	57.6	44.35

Table4.8Result for CV-syllable based.

Number of state	NO of Gaussian Mixture	Sub-word unit used	Language model	Recognizer	Word%	Accuracy%
5	12	CV-syllable	bi-gram	Hvite	57.6	44.35
5	12	Word-internal tri-phone	bi-gram	Hvite	55.46	39.86

Table4. 9 Comparisons of CV-syllable and word-internal tri-phone

The result shows that CV-syllable is perform better than word-internal tri-phone. Because syllable based is better in representational and durational stability relative to the phonemes. In addition to this since they have the long unit and least context sensitive. We come up with 2.06% word and 4% accuracy improvement using the CV-syllable based speech unit. To represent the whole syllable structure according to [24] like V,VC,CV,CVC,CVCC the corpus size that we have is a challenge to model all of them. That is why we only model the CV-syllable.

4.5.4. Increasing the corpus size

The last experiment done in this study is checking the performance of the recognizer by increasing the corpus size. In order to increase the corpus size we add spontaneous speech corpus of Bantegize [9] to spontaneous speech corpus of Adugna [10]. This brings the total corpus to3556 utterance of 60 speakers which estimates around 4 hour and 30 minute's speech data. With this experiment we didn't come up with performance improvement rather the performance is degraded. When we hear the spontaneous speech corpus of Bantegize [9] there are so many background noise, due to this reasons the feature is not extracted well when we change .WAV file format to .MFC most of the speech data is not recognized by the tool. Then when we test the recognizer performance it is reduced to 28.34% word recognition, and 26.45% accuracy recognition. The result we achieved by increasing the corpus size is presented as follows in Table 4.10

Speaker Code	Word%	Accuracy%
tst 001	38.25	34.2
tst002	26.73	24.8
tst003	29.45	27.90
tst004	33.2	31.5
tst006	31.12	29.82
*tst007	23.27	20.25
tst008	28.67	26.56
tst009	37.96	35.34
* tst010	21.06	20.96
*tst011	29.6	26.33
* tst012	27.21	25.48
tst 013	28.45	25.94
tst014	30.69	27.24
Average	28.34	26.45

Table4.10Result for increase corpus size.

As we can see from the Table 4.10 we didn't come up with performance improvement for this experimentation due to the reason that the speech corpus we try to merge has so much noise, so it is difficult to extract the exact equivalent speech wave form. When we merge the corpus size what we get is that word error rate and recognition rate is almost closer result. This indicates that when we increase the corpus the insertion error rate is relatively reduced

4.6Finding of the study

In this study we try to explore the effect of tuning techniques, cross word tri-phone, and using changing the sub word unit to CV-syllable format, and using tri-gram language model on the performance of the recognizer. What we found out is that using trigram language model, and cross word tri-phone increases the performance of the recognizer.

After a series of experimentation is held using the test data which is prepared by Adugna [10] the performance of the recognizer is tested finally, the performance of the recognizer incremented by 3.21% word recognition and 9.27% accuracy using cross word tri-phone and 4.75% correctly recognized word and 5% accuracy using trigram language model, when we compare with the result reported by Adugna [10].

The finding of this study enhances the performance of the spontaneous speech recognition in Amharic domain. However still there are a challenge that needs more attention. To increase the corpus size since spontaneous speech corpus of Adugna [10] and judiciary domain dictation application speech corpus of Bantegize [9] uses different tool the manner they prepare the file name for their audio file and for the transcription text is different, so we adjust and make them similar manually. Modeling the non-speech events are challenging; as a result, we represent them as a phone, syllable for the acoustic model and as a word in the language model. There are the same words that were uttered differently by different speakers, dialect variation. But the dictionary used is canonical, so there is a challenge to represent dialect variations.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATION

5.1 Conclusion

With this study, the possibility of improving performance for spontaneous speech recognizer for Amharic language has been explored. We conduct a series of four experiments to increase the performance for spontaneous speech of Amharic mixed test set as well as for speaker independent test set.

The training has been done first by modeling context independent mono-phones, syllables and re-estimating the models using HERest tool. In order to improve our recognizers' accuracy we have refined our models, as a result we have developed context dependent tri-phones both cross-word and word internal tri-phones from mono-phones, for syllable we construct 238 CV-syllable models then re-estimated tri-phone models as well as CV-syllable models. By starting with single Gaussian mixture, we have increased the number of Gaussian mixture in the process of refining our models and we have got better performance at 12^{th}Gaussian mixture.

The experimentation is held using four stages; first using the tuning techniques, followed by cross-word tri-phone, tri-gram language model, and then using the sub-word unit CV-syllable based, and finally, by increasing the corpus size the performance of the recognizer is tested for both speaker independent test set and mixed test set. During the course of this study the recognizer developed different acoustic model has been tested using the test set which is already prepared by Adugna [10]. Tuning the parameter of the decoder at the time of recognition brought different recognition accuracy. Some of these parameter are insertion penalty (p), grammar scale factor(s), and pruning level (t).Using trigram language model by setting the 0.2 insertion penalty 15.0 grammar scale factor and 3000 pruning level the recognizer best perform when we compare with the result reported by Adugna[10]. By using the tri-gram language model we come up with 4.75% correctly recognized word and 5% accuracy improvement for speaker independent test set.

We achieve good recognition result using trigram language model and cross word tri-phone when we compare with the other experimentation that are conducted using tuning techniques, and CV-syllables. Using cross word tri-phone we come up with 3.21% word recognition and

9.27% accuracy improvement. The unavailability of well-prepared large spontaneous speech corpus and alternative pronunciation dictionary that handle words uttered differently by different speakers affects greatly the performance of the recognizer.

5.2 Recommendation

Among the six CV-syllable structure of Amharic [24] we are trying only the CV-syllable, because in the domain of spontaneous speech recognition the corpus which is prepared so far regarding this domain are very few in numbers, small in size and transcribed manually. These have its own effect on the performance of the recognizer. Since to check the performance of the recognizer using CV-syllable structure require large size speech corpus , so it is better to prepare large size spontaneous speech corpus and making the transcription automatic.

One of the difficult features of spontaneous speech is non-speech events (dis-fluencies); therefore, handling non-speech events has a big role in recognizer's performance. We have dealt with some issues to handle them in general but still there is a room to handle them further by applying different techniques.

Since the time we have is limited and it's difficult to get the linguistic easily to work with them. We prepare canonical pronunciation dictionary, but there are the same words that are uttered differently by different speakers, dialect variation. Therefore there is a need to prepare and use alternative pronunciation dictionary in order to improve the performance of the recognizer.

The way we try the Hidden Markov Model in one way approach after training the acoustic model we directly move to test the performance of the recognizer using test set. But, rather than doing the same procedure if there are mechanisms to give feedback after testing to train the acoustic model again we may get robust acoustic model then it increase the performance of the recognizer.

Reference

[1] Lawrence Rabiner and Biing-Hwan., *Fundamentals of speech recognition*. New Jersey, U.S.A: Prentice Hall International, 1993.

[2] M.A.Anusuya and Katti.S."Speech Recognition by Machine: A Review," (IJCSIS) International Journal of Computer Science and Information Security, Vol 6, No 3, PP181 -205, 2009.

[3] WiqasGhai and Navdeep Singh, "Literature Review on Automatic Speech Recognition," *International Journal of Computer Applications*, Vol.41, No 8 , PP43-48, March 2012.

[4] Santosh.K, and M.A.Anusuya."A Review on Speech Recognition Technique," *International Journal of Computer Applications*, Vol 10, No 3 PP16–24, November 2010.

[5] Santosh.K, and M.A.Anusuya."A Review on Speech Recognition Technique," *International Journal of Computer Applications*, Vol 10, No 3, PP16–24, November 2010.

[6] Vrinda, and Chander.S, "Speech recognition system for English language," *International Journal of Advanced Research in Computer and Communication Engineering*, Vol.2, No. 1, PP 919-922, January 2013.

[7] Sanjib.D."Speech Recognition Technique: A Review," *International Journal of Engineering Research and Applications*, Vol. 2, No 3, PP 943-947, May-Jun 2012.

[8] Woosung.K. "Language model adaptation for automatic speech recognition and statistical machine translation*"*, Vol.3, No 6, PP12-17, October 2004.

[9] Bantegize Addis."*BRANA (ብራና): "Application of Amharic speech Recognition system for Dictation in judicial domain"*, MSc Thesis, University of Gondar, Ethiopia, May 2014.

[10] Adugna Deksiso."*Spontaneous Speech Recognition for Amharic Using HMM"*, MSc Thesis, Addis Ababa University, Ethiopia, March 2015.

[11] Solomon.T and Wolfgang."Syllable-Based Speech Recognition for Amharic"," in *Proceedings of the 5th Workshop on Important Unresolved Matters*, Prague, Czech Republic, June 2007, PP 33–40.

[12] Kinfe Tadesse. "S*ub-word based Amharic word recognition: an experiment using hidden Markov Model (HMM)"*.MSc Thesis, AAU, Ethiopia, June 2002.

[13] Martha Yifru."*Application of Amharic speech recognition system to command and control computer: an experiment with Microsoft word*. , MSc thesis, AAU, Ethiopia, JULY, 2003.

[14] Tuka.Al. "Lexical and Language Modeling of Diacritics and Morphemes in Arabic Automatic Speech Recognition*"*, MSc Thesis, Massachusetts Institute of Technology, February2014.

[15] Namrata.D."Feature Extraction Methods LPC, PLP and MFCC," *International journal for Advance research in Engineering and technology,* Vol 1, No VI, PP 1-5, July 2013.

[16] Jurafisky, and Martin."*Speech and language processing an introduction to Natural language processing, computational linguistics, and speech recognition"*. 2nd ed. New Jersey, U.S.A: Prentice Hall, 2007.

[17] Thangarajan. "*Word and Triphone Based Approaches in Continuous Speech,*" Vol. 4, No 3, PP 34-39, March 2008.

[18] Shuangyu Chang."A *Syllable, Articulatory-Feature, and Stress-Accent Model of Speech Recognition*", September 2002.

[19] Mohamed.m, azmi , Hesham tolba , Sherif mahdy , and mervat fashal "syllable-based automatic Arabic speech recognition," in *Proceedings of the 7th WSEAS International Conference on Signal Processing, Robotics and Automation (ISPRA '08),* UK, 2008, PP20-22.

[20] Juraj. K, "HTK vs. SPHINX for SPEECH Recognition," Vol 10, No 5, PP 202-204, 2003.

[21] Steve.Y , Gunnar Evermann, Mark Gales, Thomas Hain, Dan Kershaw, Xunying (Andrew) LiuGareth Moore, Julian Odell, Dave Ollason, Dan Povey, Valtcho Valtchev, and Phil Wood "*The HTK Book*", March 2009.

[22] AbuZeina.D, "Cross-Word Arabic Pronunciation Variation Modeling Using Part of Speech Tagging," 2009.

[23] BayeYimam, *የአማርኛሰዋሰው*. Addis Ababa: ትመማማ ድ, 2002.

[24] Sebsibe.H, "Unit Selection voice for Amharic using festvox," 5th ISCA Speech Synthesis Workshop, 2003.

[25] H. F. Ong and A. M. Ahmad, “Malay Language Speech Recognizer with Hybrid Hidden Markov Model and Artificial Neural Network “, International Journal of Information and Education Technology, Vol 1, No 2, June 2011.

[26] Muhirwe.J, “Automatic speech recognition: human computer interface forkinyarwanda Language.” Vol 4, No 3 , PP 56-63, August 2005.

[27] Gales M. and Young S.”The application of Hidden Markov Models in speech recognition”, Cambrige University, Cambrige, UK, 2008.

[28] Solomon Teferra Abate, Automatic Speech Recognition for Amharic. Ph.D. Thesis.

Available at: http://www.sub.unihamburg.de/opus/volltexte/2006/2981/pdf/thesis.pdf, 2006, accessed on 03/05/2014.

[29] André Gustavo Adami, “Automatic Speech Recognition: From the Beginning to the Portuguese Language”, Brasil,(cf), 2012.

[30] http://www.csa.gov.et/ accessed on 05/06/2015.

APPENDIX

Appendix A sample pronunciation dictionary for CV-syllable.

BR [BR] BR sp
CEbITEwI [CEbITEwI] CE bI TE wI sp
CElEmEmIbINI [CElEmEmIbINI] CE lE mE mI bI NI sp
CEle [CEle] CE le sp
CEmErIkuNI [CEmErIkuNI] CE mE rI ku NI sp
CEmIrEhI [CEmIrEhI] CE mI rE hI sp
CEmIrEwI [CEmIrEwI] CE mI rE wI sp
CEmIrIyalEwI [CEmIrIyalEwI] CE mI rI ya lE wI sp
CEmIro [CEmIro] CE mI ro sp
CErEsIku [CErEsIku] CE rE sI ku sp
CErEsuna [CErEsuna] CE rE su na sp
CErISalEhu [CErISalEhu] CE rI Sa lE hu sp
CErISalEwI [CErISalEwI] CE rI Sa lE wI sp
CErIqosI [CErIqosI] CE rI qo sI sp
CErIqosInI [CErIqosInI] CE rI qo sI nI sp
CErIqunI [CErIqunI] CE rI qu nI sp
CErIsEnI [CErIsEnI] CE rI sE nI sp
CErIsEnalI [CErIsEnalI] CE rI sE na lI sp
CErIsEwalI [CErIsEwalI] CE rI sE wa lI sp
CErIsacIhu [CErIsacIhu] CE rI sa cI hu sp
CEwanEtI [CEwanEtI] CE wa nE tI sp
CEwata [CEwata] CE wa ta sp
CIbITI [CIbITI] CI bI TI sp
CIbITacInI [CIbITacInI] CI bI Ta cI nI sp
CIbITacInInI [CIbITacInInI] CI bI Ta cI nI nI sp
CIbITunI [CIbITunI] CI bI Tu nI sp
CIbIcEbawInImI [CIbIcEbawInImI] CI bI cE ba wI nI mI sp
CIfEracEwInI [CIfEracEwInI] CI fE ra cE wI nI sp

CImIrI [CImIrI] CI mI rI sp

CInIqEtI [CInIqEtI] CI nI qE tI sp

CIno [CIno] CI no sp

CIqICIqI [CIqICIqI] CI qI CI qI sp

CIraSI [CIraSI] CI ra SI sp

CIraSImI [CIraSImI] CI ra SI mI sp

CIwIwItacInI [CIwIwItacInI] CI wI wI ta cI nI sp

CaCoho [CaCoho] Ca Co ho sp

Cama [Cama] Ca ma sp

CamamI [CamamI] Ca ma mI sp

CamawocI [CamawocI] Ca ma wo cI sp

CawEta [CawEta] Ca wE ta sp

CawEtacInInI [CawEtacInInI] Ca wE ta cI nI nI sp

DEbIdI [DEbIdI] DE bI dI sp

DEgInInEtI [DEgInInEtI] DE gI nI nE tI sp

DEgIna [DEgIna] DE gI na sp

DEgInocI [DEgInocI] DE gI no cI sp

DEgInocImI [DEgInocImI] DE gI no cI mI sp

DElIba [DElIba] DE lI ba sp

DElIbawocI [DElIbawocI] DE lI ba wo cI sp

DEmErE [DEmErE] DE mE rE sp

DEmErEcI [DEmErEcI] DE mE rE cI sp

DEmErEwI [DEmErEwI] DE mE rE wI sp

DEmErI [DEmErI] DE mE rI sp

DEmErIku [DEmErIku] DE mE rI ku sp

DEmErIkuNI [DEmErIkuNI] DE mE rI ku NI sp

DEmErIkutI [DEmErIkutI] DE mE rI ku tI sp

DEmErInI [DEmErInI] DE mE rI nI sp

DEmEru [DEmEru] DE mE ru sp

DEmIrEnalI [DEmIrEnalI] DE mI rE na lI sp

Appendix B: Sample list of audio files and their equivalent mfc files (codetrain.scp file)

audio/tr001.001.wav mfcc/tr001.001.mfc

audio/tr001.002.wav mfcc/tr001.002.mfc

audio/tr001.003.wav mfcc/tr001.003.mfc

audio/tr001.004.wav mfcc/tr001.004.mfc

audio/tr001.005.wav mfcc/tr001.005.mfc

audio/tr001.006.wav mfcc/tr001.006.mfc

audio/tr001.007.wav mfcc/tr001.007.mfc

audio/tr001.008.wav mfcc/tr001.008.mfc

audio/tr001.009.wav mfcc/tr001.009.mfc

audio/tr001.010.wav mfcc/tr001.010.mfc

…..

audio/tr036.090.wav mfcc/tr036.090.mfc

audio/tr036.091.wav mfcc/tr036.091.mfc

audio/tr036.092.wav mfcc/tr036.092.mfc

audio/tr036.093.wav mfcc/tr036.093.mfc

audio/tr036.094.wav mfcc/tr036.094.mfc

audio/tr036.095.wav mfcc/tr036.095.mfc

audio/tr036.096.wav mfcc/tr036.096.mfc

audio/tr036.097.wav mfcc/tr036.097.mfc

Appendix C: Configuration files

HDECode tool config file

TARGETRATE = 100000.0

SAVECOMPRESSED= TRUE

SAVEWITHCRC = TRUE

WINDOWSIZE = 250000.0

USEHAMMING = TRUE

PREEMCOEF = 0.97

NUMCHANS = 26

CEPLIFTER = 22

NUMCEPS = 12

USESILDET = TRUE

OUTSILWARN = TRUE

MICIN = TRUE

SOURCERATE = 625

RAWMITFORMAT = TRUE

ALLOWXWRDEXP = TRUE

FORCECXTEXP = TRUE

TARGETKIND = MFCC_0_D_A

STARTWORD = SENT-START

ENDWORD = SENT-END

HCopy tool config file

TARGETKIND = MFCC_0_D_A

SOURCEKIND = WAV

SOURCEFORMAT = WAV

TARGETFORMAT = HTK

SOURCERATE=625

TARGETRATE = 100000.0

SAVECOMPRESSED = TRUE

SAVEWITHCRC = TRUE

WINDOWSIZE = 250000.0

USEHAMMING = TRUE

PREEMCOEF = 0.97

NUMCHANS = 26

CEPLIFTER = 22

NUMCEPS = 12

ENORMALISE = FALSE

HCompV tool config file

TARGETKIND = MFCC_0_D_A

SOURCEFORMAT = HTK

SOURCERATE = 625

TARGETRATE = 100000.0

SAVECOMPRESSED = TRUE

SAVEWITHCRC = TRUE

WINDOWSIZE = 250000.0

USEHAMMING = TRUE

PREEMCOEF = 0.97

NUMCHANS = 26

CEPLIFTER = 22

NUMCEPS = 12

ENORMALISE=FALSE

Appendix E:Proto file for 5 emitting state HMM

~o <VecSize> 39 <MFCC_0_D_A>

~h "proto"

<BeginHMM>

<NumStates> 7

<State> 2 <NumMixes> 1

<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 …

<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 …

<State> 3 <NumMixes> 1

<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 …

<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 …

<State> 4 <NumMixes> 1

<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 …

<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 …

<State> 5 <NumMixes> 1

<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…

<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 …

<State> 6 <NumMixes> 1 95

<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…

<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…

<TransP> 7

0.0 1.0 0.0 0.0 0.0 0.0 0.0

0.0 0.6 0.4 0.0 0.0 0.0 0.0

0.0 0.0 0.6 0.4 0.0 0.0 0.0

0.0 0.0 0.0 0.6 0.4 0.0 0.0

0.0 0.0 0.0 0.0 0.6 0.4 0.0

0.0 0.0 0.0 0.0 0.0 0.6 0.4

0.0 0.0 0.0 0.0 0.0 0.0 0.0

<EndHMM>

Appendix F: Editing script tree.hed

```
RO 100.0 stats
TR 0
QS "R_NonBoundary" { "*+*" }
QS "R_Silence" { "*+sil" }
QS "R_Stop" { "*+C","*+g","*+k","*+d","*+b","*+t","*+P","*+p" }
QS "R_Nasal" { "*+n","*+m" }
QS "R_Fricative" { "*+s" }
QS "R_Liquid" { "*+y","*+w","*+r","*+l" }
QS "R_Vowel" { "*+I","*+E","*+a","*+u","*+o","*+i","*+e" }
QS "R_C-Front" { "*+w","*+m","*+b","*+P","*+p" }
QS "R_C-Central" { "*+r","*+l","*+S","*+z","*+s","*+n","*+d","*+t" }
QS "R_C-Back" { "*+g","*+k","*+y" }
QS "R_V-Front" { "*+e","*+i" }
QS "R_V-Central" { "*+I","*+a","*+E" }
QS "R_V-Back" { "*+o","*+u" }
QS "R_Front" { "*+e","*+i","*+w","*+m","*+b" }
QS "R_Central" { "*+I","*+E","*+a","*+r","*+l","*+S","*+s","*+n","*+d","*+t" }
QS "R_Back" { "*+u","*+o","*+g","*+k","*+y","*+T","*+S" }
QS "R_Fortis" { "*+S","*+s","*+k","*+t","*+p" }
QS "R_Lenis" { "*+g","*+d","*+b" }
QS "R_UnFortLenis" { "*+w","*+y","*+r","*+l","*+n","*+m" }
QS "R_Coronal" { "*+r","*+l","*+s","*+n","*+d","*+t" }
QS "R_NonCoronal" { "*+w","*+y","*+g","*+k","*+m","*+b","*+p" }
QS "R_Anterior" { "*+w","*+l","*+s","*+n","*+d","*+t","*+m","*+b","*+p" }
QS "R_NonAnterior" { "*+y","*+r","*+g","*+k" }
QS "R_Continuent" { "*+w","*+y","*+r","*+l","*+S","*+s","*+n","*+m" } 97
```

QS "R_NonContinuent" { "*+c","*+g","*+k","*+d","*+t","*+b","*+p" }

QS "R_Strident" { "*+c","*+S","*+s" }

QS "R_Glide" { "*+w","*+y","*+r","*+l" }

QS "R_Syllabic" { "*+l","*+m" }

QS "R_Unvoiced-Cons" { "*+C","*+s","*+k","*+t" }

QS "R_Voiced-Cons" { "*+z","*+w","*+r","*+n","*+m","*+l","*+y","*+g","*+d","*+b" }

QS "R_Unvoiced-All" { "*+sil","*+s","*+k","*+t" }

QS "R_Long" { "*+l","*+a" }

QS "R_Fronting" { "*+i","*+e" }

QS "R_High" { "*+u","*+I","*+i" }

QS "R_Medium" { "*+l","*+m","*+o","*+E","*+e" }

QS "R_Rounded" { "*+w","*+o","*+u" }

QS "R_Unrounded" { "*+y","*+r","*+l","*+E","*+I","*+i","*+e","*+a" }

QS "R_NonAffricate" { "*+s" }

QS "R_Affricate" { "*+c" }

QS "R_IVowel" { "*+i" }

QS "R_EVowel" { "*+e","*+E" }

QS "R_AVowel" { "*+a" }

QS "R_OVowel" { "*+o" }

QS "R_UVowel" { "*+u","*+l","*+m","*+a" }

QS "R_Voiced-Stop" { "*+D","*+g","*+d","*+b" }

QS "R_Unvoiced-Stop" { "*+H","*+k","*+c","*+t","*+p" }

QS "R_Front-Stop" { "*+b" }

QS "R_Central-Stop" { "*+d","*+t" }

QS "R_Back-Stop" { "*+g","*+k" }

QS "R_Voiced-Fric" { "*+z" }

QS "R_Unvoiced-Fric" { "*+c","*+f","*+S","*+s" } 98

```
QS "R_Central-Fric" { "*+z","*+s" }
QS "R_a" { "*+a" }
QS "R_b" { "*+b" }
QS "R_c" { "*+c" }
QS "R_C" { "*+C" }
QS "R_d" { "*+d" }
QS "R_D" { "*+D" }
QS "R_e" { "*+e" }
QS "R_E" { "*+E" }
QS "R_f" { "*+f" }
QS "R_g" { "*+g" }
QS "R_H" { "*+H" }
QS "R_h" { "*+h" }
QS "R_I" { "*+I" }
QS "R_i" { "*+i" }
QS "R_k" { "*+k" }
QS "R_l" { "*+l" }
QS "R_m" { "*+m" }
QS "R_n" { "*+n" }
QS "R_N" { "*+N" }
QS "R_o" { "*+o" }
QS "R_P" { "*+P" }
QS "R_p" { "*+p" }
QS "R_q" { "*+q" }
QS "R_r" { "*+r" }
QS "R_s" { "*+s" }
QS "R_S" { "*+S" }
QS "R_t" { "*+t" }
QS "R_T" { "*+T" }
QS "R_u" { "*+u" }
```

QS "R_v" { "*+v" }

QS "R_w" { "*+w" }

QS "R_x" { "*+x" }

QS "R_y" { "*+y" }

QS "R_z" { "*+z" }

QS "R_Z" { "*+Z" }

QS "R_BR" { "*+BR" }

QS "R_FP" { "*+FP" }

QS "R_HES" { "*+HES" }

QS "R_LIP" { "*+LIP" }

QS "R_OTH" { "*+OTH" }

QS "R_THC" { "*+THC" }

QS "R_LGH" { "*+LGH" }

QS "R_ua" { "*+ua" }

QS "R_sil" { "*+sil" }

QS "L_NonBoundary" { "*-*" }

QS "L_Silence" { "sil-*" }

QS "L_Stop" { "C-*","g-*","k-*","d-*","b-*","t-*","P-*","p-*" }

QS "L_Nasal" { "n-*","m-*" } 99

QS "L_Fricative" { "s-*" }

QS "L_Liquid" { "y-*","w-*","r-*","l-*" }

QS "L_Vowel" { "I-*","E-*","a-*","u-*","o-*","i-*","e-*" }

QS "L_C-Front" { "w-*","m-*","b-*","P-*","p-*" }

QS "L_C-Central" { "r-*","l-*","S-*","z-*","s-*","n-*","d-*","t-*" }

QS "L_C-Back" { "g-*","k-*","y-*" }

QS "L_V-Front" { "e-*","i-*" }

QS "L_V-Central" { "I-*","a-*","E-*" }

QS "L_V-Back" { "o-*","u-*" }

QS "L_Front" { "e-*","i-*","w-*","m-*","b-*" }

QS "L_Central" { "I-*","E-*","a-*","r-*","l-*","S-*","s-*","n-*","d-*","t-*" }

QS "L_Back" { "u-*","o-*","g-*","k-*","y-*","T-*","S-*" }

QS "L_Fortis" { "S-*","s-*","k-*","t-*","p-*" }

QS "L_Lenis" { "g-*","d-*","b-*" }

QS "L_UnFortLenis" { "w-*","y-*","r-*","l-*","n-*","m-*" }

QS "L_Coronal" { "r-*","l-*","s-*","n-*","d-*","t-*" }

QS "L_NonCoronal" { "w-*","y-*","g-*","k-*","m-*","b-*","p-*" }

QS "L_Anterior" { "w-*","l-*","s-*","n-*","d-*","t-*","m-*","b-*","p-*" }

QS "L_NonAnterior" { "y-*","r-*","g-*","k-*" }

QS "L_Continuent" { "w-*","y-*","r-*","l-*","S-*","s-*","n-*","m-*" }

QS "L_NonContinuent" { "c-*","g-*","k-*","d-*","t-*","b-*","p-*" }

QS "L_Strident" { "c-*","S-*","s-*" }

QS "L_Glide" { "w-*","y-*","r-*","l-*" }

QS "L_Syllabic" { "l-*","m-*" }

QS "L_Unvoiced-Cons" { "C-*","s-*","k-*","t-*" }

QS "L_Voiced-Cons" { "z-*","w-*","r-*","n-*","m-*","l-*","y-*","g-*","d-*","b-*" }

QS "L_Unvoiced-All" { "sil-*","s-*","k-*","t-*" } 100

QS "L_Long" { "l-*","a-*" }

QS "L_Fronting" { "i-*","e-*" }

QS "L_High" { "u-*","I-*","i-*" }

QS "L_Medium" { "l-*","m-*","o-*","E-*","e-*" }

QS "L_Rounded" { "w-*","o-*","u-*" }

QS "L_Unrounded" { "y-*","r-*","l-*","E-*","I-*","i-*","e-*","a-*" }

QS "L_NonAffricate" { "s-*" }

QS "L_Affricate" { "c-*" }

QS "L_IVowel" { "i-*" }

QS "L_EVowel" { "e-*","E-*" }

QS "L_AVowel" { "a-*" }

QS "L_OVowel" { "o-*" }

QS "L_UVowel" { "u-*","l-*","m-*","a-*" }

QS "L_Voiced-Stop" { "D-*","g-*","d-*","b-*" }

QS "L_Unvoiced-Stop" { "H-*","k-*","c-*","t-*","p-*" }

QS "L_Front-Stop" { "b-*" }

QS "L_Central-Stop" { "d-*","t-*" }

QS "L_Back-Stop" { "g-*","k-*" }

QS "L_Voiced-Fric" { "z-*" }

QS "L_Unvoiced-Fric" { "c-*","f-*","S-*","s-*" }

QS "L_Central-Fric" { "z-*","s-*" }

QS "L_a" { "a-*" }

QS "L_b" { "b-*" }

QS "L_c" { "c-*" }

QS "L_C" { "C-*" }

QS "L_d" { "d-*" }

QS "L_D" { "D-*" }

QS "L_e" { "e-*" }

QS "L_E" { "E-*" }

QS "L_f" { "f-*" }

QS "L_g" { "g-*" }

QS "L_H" { "H-*" }
QS "L_h" { "h-*" } 101

QS "L_I" { "I-*" }

QS "L_i" { "i-*" }

QS "L_k" { "k-*" }

QS "L_l" { "l-*" }

QS "L_m" { "m-*" }

QS "L_n" { "n-*" }

QS "L_N" { "N-*" }

QS "L_o" { "o-*" }

QS "L_p" { "p-*" }

QS "L_P" { "P-*" }

QS "L_q" { "q-*" }

QS "L_r" { "r-*" }

QS "L_s" { "s-*" }

QS "L_S" { "S-*" }

QS "L_t" { "t-*" }

QS "L_T" { "T-*" }

QS "L_u" { "u-*" }

QS "L_w" { "w-*" }

QS "L_x" { "x-*" }

QS "L_y" { "y-*" }

QS "L_z" { "z-*" }

QS "L_Z" { "Z-*" }

QS "L_v" { "v-*" }

QS "L_LIP" { "LIP-*" }

QS "L_ua" { "ua-*" }

QS "L_THC" { "THC-*" }

QS "L_FP" { "FP-*" }

QS "L_BR" { "BR-*" }

QS "L_HES" { "HES-*" }

QS "L_OTH" { "OTH-*" }

QS "L_LGH" { "LGH-*" }

```
QS "L_sil" { "sil-*" }
TR 2
TB 600 "ST_BR_2_" {("BR","*-BR+*","BR+*","*-BR").state[2]}
TB 600 "ST_C_2_" {("C","*-C+*","C+*","*-C").state[2]}
TB 600 "ST_E_2_" {("E","*-E+*","E+*","*-E").state[2]}
TB 600 "ST_b_2_" {("b","*-b+*","b+*","*-b").state[2]}
TB 600 "ST_I_2_" {("I","*-I+*","I+*","*-I").state[2]}
TB 600 "ST_T_2_" {("T","*-T+*","T+*","*-T").state[2]}
TB 600 "ST_w_2_" {("w","*-w+*","w+*","*-w").state[2]}
TB 600 "ST_l_2_" {("l","*-l+*","l+*","*-l").state[2]}
TB 600 "ST_m_2_" {("m","*-m+*","m+*","*-m").state[2]}
TB 600 "ST_N_2_" {("N","*-N+*","N+*","*-N").state[2]} 102
```

TB 600 "ST_e_2_" {("e","*-e+*","e+*","*-e").state[2]}

TB 600 "ST_r_2_" {("r","*-r+*","r+*","*-r").state[2]}

TB 600 "ST_k_2_" {("k","*-k+*","k+*","*-k").state[2]}

TB 600 "ST_u_2_" {("u","*-u+*","u+*","*-u").state[2]}

TB 600 "ST_h_2_" {("h","*-h+*","h+*","*-h").state[2]}

TB 600 "ST_y_2_" {("y","*-y+*","y+*","*-y").state[2]}

TB 600 "ST_a_2_" {("a","*-a+*","a+*","*-a").state[2]}

TB 600 "ST_s_2_" {("s","*-s+*","s+*","*-s").state[2]}

TB 600 "ST_n_2_" {("n","*-n+*","n+*","*-n").state[2]}

TB 600 "ST_S_2_" {("S","*-S+*","S+*","*-S").state[2]}

TB 600 "ST_q_2_" {("q","*-q+*","q+*","*-q").state[2]}

TB 600 "ST_o_2_" {("o","*-o+*","o+*","*-o").state[2]}

TB 600 "ST_t_2_" {("t","*-t+*","t+*","*-t").state[2]}

TB 600 "ST_c_2_" {("c","*-c+*","c+*","*-c").state[2]}

TB 600 "ST_f_2_" {("f","*-f+*","f+*","*-f").state[2]}

TB 600 "ST_D_2_" {("D","*-D+*","D+*","*-D").state[2]}

TB 600 "ST_d_2_" {("d","*-d+*","d+*","*-d").state[2]}

TB 600 "ST_g_2_" {("g","*-g+*","g+*","*-g").state[2]}

TB 600 "ST_i_2_" {("i","*-i+*","i+*","*-i").state[2]}

TB 600 "ST_ua_2_" {("ua","*-ua+*","ua+*","*-ua").state[2]}

TB 600 "ST_p_2_" {("p","*-p+*","p+*","*-p").state[2]}

TB 600 "ST_v_2_" {("v","*-v+*","v+*","*-v").state[2]}

TB 600 "ST_FP_2_" {("FP","*-FP+*","FP+*","*-FP").state[2]}

TB 600 "ST_HES_2_" {("HES","*-HES+*","HES+*","*-HES").state[2]}

TB 600 "ST_H_2_" {("H","*-H+*","H+*","*-H").state[2]}

TB 600 "ST_z_2_" {("z","*-z+*","z+*","*-z").state[2]}

TB 600 "ST_x_2_" {("x","*-x+*","x+*","*-x").state[2]} 103

TB 600 "ST_Z_2_" {("Z","*-Z+*","Z+*","*-Z").state[2]}

TB 600 "ST_P_2_" {("P","*-P+*","P+*","*-P").state[2]}

TB 600 "ST_OTH_2_" {("OTH","*-OTH+*","OTH+*","*-OTH").state[2]}

TB 600 "ST_sil_2_" {("sil","*-sil+*","sil+*","*-sil").state[2]}

TB 600 "ST_BR_3_" {("BR","*-BR+*","BR+*","*-BR").state[3]}

TB 600 "ST_C_3_" {("C","*-C+*","C+*","*-C").state[3]}

TB 600 "ST_E_3_" {("E","*-E+*","E+*","*-E").state[3]}

TB 600 "ST_b_3_" {("b","*-b+*","b+*","*-b").state[3]}

TB 600 "ST_I_3_" {("I","*-I+*","I+*","*-I").state[3]}

TB 600 "ST_T_3_" {("T","*-T+*","T+*","*-T").state[3]}

TB 600 "ST_w_3_" {("w","*-w+*","w+*","*-w").state[3]}

TB 600 "ST_l_3_" {("l","*-l+*","l+*","*-l").state[3]}

TB 600 "ST_m_3_" {("m","*-m+*","m+*","*-m").state[3]}

TB 600 "ST_N_3_" {("N","*-N+*","N+*","*-N").state[3]}

TB 600 "ST_e_3_" {("e","*-e+*","e+*","*-e").state[3]}

TB 600 "ST_r_3_" {("r","*-r+*","r+*","*-r").state[3]}

TB 600 "ST_k_3_" {("k","*-k+*","k+*","*-k").state[3]}

TB 600 "ST_u_3_" {("u","*-u+*","u+*","*-u").state[3]}

TB 600 "ST_h_3_" {("h","*-h+*","h+*","*-h").state[3]}

TB 600 "ST_y_3_" {("y","*-y+*","y+*","*-y").state[3]}

TB 600 "ST_a_3_" {("a","*-a+*","a+*","*-a").state[3]}

TB 600 "ST_s_3_" {("s","*-s+*","s+*","*-s").state[3]}

TB 600 "ST_n_3_" {("n","*-n+*","n+*","*-n").state[3]}

TB 600 "ST_S_3_" {("S","*-S+*","S+*","*-S").state[3]}

TB 600 "ST_q_3_" {("q","*-q+*","q+*","*-q").state[3]}

TB 600 "ST_o_3_" {("o","*-o+*","o+*","*-o").state[3]}

TB 600 "ST_t_3_" {("t","*-t+*","t+*","*-t").state[3]} 104

TB 600 "ST_c_3_" {("c","*-c+*","c+*","*-c").state[3]}

TB 600 "ST_f_3_" {("f","*-f+*","f+*","*-f").state[3]}

TB 600 "ST_D_3_" {("D","*-D+*","D+*","*-D").state[3]}

TB 600 "ST_d_3_" {("d","*-d+*","d+*","*-d").state[3]}

TB 600 "ST_g_3_" {("g","*-g+*","g+*","*-g").state[3]}

TB 600 "ST_i_3_" {("i","*-i+*","i+*","*-i").state[3]}

TB 600 "ST_ua_3_" {("ua","*-ua+*","ua+*","*-ua").state[3]}

TB 600 "ST_p_3_" {("p","*-p+*","p+*","*-p").state[3]}

TB 600 "ST_v_3_" {("v","*-v+*","v+*","*-v").state[3]}

TB 600 "ST_FP_3_" {("FP","*-FP+*","FP+*","*-FP").state[3]}

TB 600 "ST_HES_3_" {("HES","*-HES+*","HES+*","*-HES").state[3]}

TB 600 "ST_H_3_" {("H","*-H+*","H+*","*-H").state[3]}

TB 600 "ST_z_3_" {("z","*-z+*","z+*","*-z").state[3]}

TB 600 "ST_x_3_" {("x","*-x+*","x+*","*-x").state[3]}

TB 600 "ST_Z_3_" {("Z","*-Z+*","Z+*","*-Z").state[3]}

TB 600 "ST_P_3_" {("P","*-P+*","P+*","*-P").state[3]}

TB 600 "ST_OTH_3_" {("OTH","*-OTH+*","OTH+*","*-OTH").state[3]}

TB 600 "ST_sil_3_" {("sil","*-sil+*","sil+*","*-sil").state[3]}

TB 600 "ST_BR_4_" {("BR","*-BR+*","BR+*","*-BR").state[4]}

TB 600 "ST_C_4_" {("C","*-C+*","C+*","*-C").state[4]}

TB 600 "ST_E_4_" {("E","*-E+*","E+*","*-E").state[4]}

TB 600 "ST_b_4_" {("b","*-b+*","b+*","*-b").state[4]}

TB 600 "ST_I_4_" {("I","*-I+*","I+*","*-I").state[4]}

TB 600 "ST_T_4_" {("T","*-T+*","T+*","*-T").state[4]}

TB 600 "ST_w_4_" {("w","*-w+*","w+*","*-w").state[4]}

TB 600 "ST_l_4_" {("l","*-l+*","l+*","*-l").state[4]}

TB 600 "ST_m_4_" {("m","*-m+*","m+*","*-m").state[4]} 105

TB 600 "ST_N_4_" {("N","*-N+*","N+*","*-N").state[4]}

TB 600 "ST_e_4_" {("e","*-e+*","e+*","*-e").state[4]}

TB 600 "ST_r_4_" {("r","*-r+*","r+*","*-r").state[4]}

TB 600 "ST_k_4_" {("k","*-k+*","k+*","*-k").state[4]}

TB 600 "ST_u_4_" {("u","*-u+*","u+*","*-u").state[4]}

TB 600 "ST_h_4_" {("h","*-h+*","h+*","*-h").state[4]}

TB 600 "ST_y_4_" {("y","*-y+*","y+*","*-y").state[4]}

TB 600 "ST_a_4_" {("a","*-a+*","a+*","*-a").state[4]}

TB 600 "ST_s_4_" {("s","*-s+*","s+*","*-s").state[4]}

TB 600 "ST_n_4_" {("n","*-n+*","n+*","*-n").state[4]}

TB 600 "ST_S_4_" {("S","*-S+*","S+*","*-S").state[4]}

TB 600 "ST_q_4_" {("q","*-q+*","q+*","*-q").state[4]}

TB 600 "ST_o_4_" {("o","*-o+*","o+*","*-o").state[4]}

TB 600 "ST_t_4_" {("t","*-t+*","t+*","*-t").state[4]}

TB 600 "ST_c_4_" {("c","*-c+*","c+*","*-c").state[4]}

TB 600 "ST_f_4_" {("f","*-f+*","f+*","*-f").state[4]}

TB 600 "ST_D_4_" {("D","*-D+*","D+*","*-D").state[4]}

TB 600 "ST_d_4_" {("d","*-d+*","d+*","*-d").state[4]}

TB 600 "ST_g_4_" {("g","*-g+*","g+*","*-g").state[4]}

TB 600 "ST_i_4_" {("i","*-i+*","i+*","*-i").state[4]}

TB 600 "ST_ua_4_" {("ua","*-ua+*","ua+*","*-ua").state[4]}

TB 600 "ST_p_4_" {("p","*-p+*","p+*","*-p").state[4]}

TB 600 "ST_v_4_" {("v","*-v+*","v+*","*-v").state[4]}

TB 600 "ST_FP_4_" {("FP","*-FP+*","FP+*","*-FP").state[4]}

TB 600 "ST_HES_4_" {("HES","*-HES+*","HES+*","*-HES").state[4]}

TB 600 "ST_H_4_" {("H","*-H+*","H+*","*-H").state[4]}

TB 600 "ST_z_4_" {("z","*-z+*","z+*","*-z").state[4]} 106

TB 600 "ST_x_4_" {("x","*-x+*","x+*","*-x").state[4]}

TB 600 "ST_Z_4_" {("Z","*-Z+*","Z+*","*-Z").state[4]}

TB 600 "ST_P_4_" {("P","*-P+*","P+*","*-P").state[4]}

TB 600 "ST_OTH_4_" {("OTH","*-OTH+*","OTH+*","*-OTH").state[4]}

TB 600 "ST_sil_4_" {("sil","*-sil+*","sil+*","*-sil").state[4]}

TB 600 "ST_BR_5_" {("BR","*-BR+*","BR+*","*-BR").state[5]}

TB 600 "ST_C_5_" {("C","*-C+*","C+*","*-C").state[5]}

TB 600 "ST_E_5_" {("E","*-E+*","E+*","*-E").state[5]}

TB 600 "ST_b_5_" {("b","*-b+*","b+*","*-b").state[5]}

TB 600 "ST_I_5_" {("I","*-I+*","I+*","*-I").state[5]}

TB 600 "ST_T_5_" {("T","*-T+*","T+*","*-T").state[5]}

TB 600 "ST_w_5_" {("w","*-w+*","w+*","*-w").state[5]}

TB 600 "ST_l_5_" {("l","*-l+*","l+*","*-l").state[5]}

TB 600 "ST_m_5_" {("m","*-m+*","m+*","*-m").state[5]}

TB 600 "ST_N_5_" {("N","*-N+*","N+*","*-N").state[5]}

TB 600 "ST_e_5_" {("e","*-e+*","e+*","*-e").state[5]}

TB 600 "ST_r_5_" {("r","*-r+*","r+*","*-r").state[5]}

TB 600 "ST_k_5_" {("k","*-k+*","k+*","*-k").state[5]}

TB 600 "ST_u_5_" {("u","*-u+*","u+*","*-u").state[5]}

TB 600 "ST_h_5_" {("h","*-h+*","h+*","*-h").state[5]}

TB 600 "ST_y_5_" {("y","*-y+*","y+*","*-y").state[5]}

TB 600 "ST_a_5_" {("a","*-a+*","a+*","*-a").state[5]}

TB 600 "ST_s_5_" {("s","*-s+*","s+*","*-s").state[5]}

TB 600 "ST_n_5_" {("n","*-n+*","n+*","*-n").state[5]}

TB 600 "ST_S_5_" {("S","*-S+*","S+*","*-S").state[5]}

TB 600 "ST_q_5_" {("q","*-q+*","q+*","*-q").state[5]}

TB 600 "ST_o_5_" {("o","*-o+*","o+*","*-o").state[5]} 107

TB 600 "ST_t_5_" {("t","*-t+*","t+*","*-t").state[5]}

TB 600 "ST_c_5_" {("c","*-c+*","c+*","*-c").state[5]}

TB 600 "ST_f_5_" {("f","*-f+*","f+*","*-f").state[5]}

TB 600 "ST_D_5_" {("D","*-D+*","D+*","*-D").state[5]}

TB 600 "ST_d_5_" {("d","*-d+*","d+*","*-d").state[5]}

TB 600 "ST_g_5_" {("g","*-g+*","g+*","*-g").state[5]}

TB 600 "ST_i_5_" {("i","*-i+*","i+*","*-i").state[5]}

TB 600 "ST_ua_5_" {("ua","*-ua+*","ua+*","*-ua").state[5]}

TB 600 "ST_p_5_" {("p","*-p+*","p+*","*-p").state[5]}

TB 600 "ST_v_5_" {("v","*-v+*","v+*","*-v").state[5]}

TB 600 "ST_FP_5_" {("FP","*-FP+*","FP+*","*-FP").state[5]}

TB 600 "ST_HES_5_" {("HES","*-HES+*","HES+*","*-HES").state[5]}

TB 600 "ST_H_5_" {("H","*-H+*","H+*","*-H").state[5]}

TB 600 "ST_z_5_" {("z","*-z+*","z+*","*-z").state[5]}

TB 600 "ST_x_5_" {("x","*-x+*","x+*","*-x").state[5]}

TB 600 "ST_Z_5_" {("Z","*-Z+*","Z+*","*-Z").state[5]}

TB 600 "ST_P_5_" {("P","*-P+*","P+*","*-P").state[5]}

TB 600 "ST_OTH_5_" {("OTH","*-OTH+*","OTH+*","*-OTH").state[5]}

TB 600 "ST_sil_5_" {("sil","*-sil+*","sil+*","*-sil").state[5]}

TB 600 "ST_BR_6_" {("BR","*-BR+*","BR+*","*-BR").state[6]}

TB 600 "ST_C_6_" {("C","*-C+*","C+*","*-C").state[6]}

TB 600 "ST_E_6_" {("E","*-E+*","E+*","*-E").state[6]}

TB 600 "ST_b_6_" {("b","*-b+*","b+*","*-b").state[6]}

TB 600 "ST_I_6_" {("I","*-I+*","I+*","*-I").state[6]}

TB 600 "ST_T_6_" {("T","*-T+*","T+*","*-T").state[6]}

TB 600 "ST_w_6_" {("w","*-w+*","w+*","*-w").state[6]}

TB 600 "ST_l_6_" {("l","*-l+*","l+*","*-l").state[6]} 108

TB 600 "ST_m_6_" {("m","*-m+*","m+*","*-m").state[6]}

TB 600 "ST_N_6_" {("N","*-N+*","N+*","*-N").state[6]}

TB 600 "ST_e_6_" {("e","*-e+*","e+*","*-e").state[6]}

TB 600 "ST_r_6_" {("r","*-r+*","r+*","*-r").state[6]}

TB 600 "ST_k_6_" {("k","*-k+*","k+*","*-k").state[6]}

TB 600 "ST_u_6_" {("u","*-u+*","u+*","*-u").state[6]}

TB 600 "ST_h_6_" {("h","*-h+*","h+*","*-h").state[6]}

TB 600 "ST_y_6_" {("y","*-y+*","y+*","*-y").state[6]}

TB 600 "ST_a_6_" {("a","*-a+*","a+*","*-a").state[6]}

TB 600 "ST_s_6_" {("s","*-s+*","s+*","*-s").state[6]}

TB 600 "ST_n_6_" {("n","*-n+*","n+*","*-n").state[6]}

TB 600 "ST_S_6_" {("S","*-S+*","S+*","*-S").state[6]}

TB 600 "ST_q_6_" {("q","*-q+*","q+*","*-q").state[6]}

TB 600 "ST_o_6_" {("o","*-o+*","o+*","*-o").state[6]}

TB 600 "ST_t_6_" {("t","*-t+*","t+*","*-t").state[6]}

TB 600 "ST_c_6_" {("c","*-c+*","c+*","*-c").state[6]}

TB 600 "ST_f_6_" {("f","*-f+*","f+*","*-f").state[6]}

TB 600 "ST_D_6_" {("D","*-D+*","D+*","*-D").state[6]}

TB 600 "ST_d_6_" {("d","*-d+*","d+*","*-d").state[6]}

TB 600 "ST_g_6_" {("g","*-g+*","g+*","*-g").state[6]}

TB 600 "ST_i_6_" {("i","*-i+*","i+*","*-i").state[6]}

TB 600 "ST_ua_6_" {("ua","*-ua+*","ua+*","*-ua").state[6]}

TB 600 "ST_p_6_" {("p","*-p+*","p+*","*-p").state[6]}

TB 600 "ST_v_6_" {("v","*-v+*","v+*","*-v").state[6]}

TB 600 "ST_FP_6_" {("FP","*-FP+*","FP+*","*-FP").state[6]}

TB 600 "ST_HES_6_" {("HES","*-HES+*","HES+*","*-HES").state[6]}

TB 600 "ST_H_6_" {("H","*-H+*","H+*","*-H").state[6]} 109

TB 600 "ST_z_6_" {("z","*-z+*","z+*","*-z").state[6]}

TB 600 "ST_x_6_" {("x","*-x+*","x+*","*-x").state[6]}

TB 600 "ST_Z_6_" {("Z","*-Z+*","Z+*","*-Z").state[6]}

TB 600 "ST_P_6_" {("P","*-P+*","P+*","*-P").state[6]}

TB 600 "ST_OTH_6_" {("OTH","*-OTH+*","OTH+*","*-OTH").state[6]}

TB 600 "ST_sil_6_" {("sil","*-sil+*","sil+*","*-sil").state[6]}

TR 1

AU "withoutnone/fulllist"

CO "withoutnone/tiedlist"

ST "withoutnone/Trees"

APPENDIX G commands used to create cross-word tri-phone for HDecode

1. Creating the xwrd and xwrd.mlf file by using the following command (HLED)

HLEd -l '*' -n xwrd/xwrd -i xwrd/xwrd.mlfxwrd/xwrd.hled phones1.mlf

//The clnone file which consist of the following command is created by name clone2.hed

MM "trP_" { *.transP }

CL "xwrd"

// creating the model file for hmm1 from hmm0 using the following command

HHEd -B -T 1 -H xwrd/hmm0/models -M xwrd/hmm1 xwrd/clone2.hed xwrd/monophones0

// using the fulllist_prl script crating full list for cross-word triphone using the following command(unseen)

perlfull_list.prl monophones0 > unseen

//Use the mkclscript to create the HHEd script

perlmkclscript TB 600.0 mono.list>>treeedit.hed

//Creating model file in the hmm1 from hmmo models based on the follwing command

HHEd -A -T 1 -H xwrd/hmm0/models -M xwrd/hmm1 xwrd/clone2.hed xwrd/monophones0

// creating models in hmm2 from hmm1 models based on the following command

HERest -A -D -T 1 -C config -I xwrd/xwrd.mlf -t 250.0 150.0 3000.0 -s xwrd/xwrdstats -S train.scp -H xwrd/hmm1/models -M xwrd/hmm2 xwrd/xwrd

// creating model in hmm3 from hmm2 model based on the following command

HERest -A -D -T 1 -C config -I xwrd/xwrd.mlf -t 250.0 150.0 3000.0 -s xwrd/xwrdstats -S train.scp -H xwrd/hmm2/models -M xwrd/hmm3 xwrd/xwrd

// creating model in hmm4 from hmm3 based on the following command

HERest -A -D -T 1 -C config -I xwrd/xwrd.mlf -t 250.0 150.0 3000.0 -s xwrd/xwrdstats -S train.scp -H xwrd/hmm3/models -M xwrd/hmm4 xwrd/xwrd

//creating model in hmm5 from hmm4 based on the following command (in addition it creates tress, treeg and clog)

HHEd -T 1 -H xwrd/hmm4/models -M xwrd/hmm5 xwrd/clusterFinal.hedxwrd/xwrd>xwrd/clog

//creating model in hmm6 from hmmsbasd on the following command

HERest -A -D -T 1 -C config -I xwrd/xwrd.mlf -t 250.0 150.0 3000.0 -s xwrd/xwrdstats -S train.scp -H xwrd/hmm5/models -M xwrd/hmm6 xwrd/treeg

//creating model in hmm7 from hmm6 based on the following command

HERest -A -D -T 1 -C config -I xwrd/xwrd.mlf -t 250.0 150.0 3000.0 -s xwrd/xwrdstats -S train.scp -H xwrd/hmm6/models -M xwrd/hmm7 xwrd/treeg

//creating model in hmm8 from hmm7 based on the following command

HERest -A -D -T 1 -C config -I xwrd/xwrd.mlf -t 250.0 150.0 3000.0 -s xwrd/xwrdstats -S train.scp -H xwrd/hmm7/models -M xwrd/hmm8 xwrd/treeg

//creating model in hmm9 from hmm8 based on the following command

HERest -A -D -T 1 -C config -I xwrd/xwrd.mlf -t 250.0 150.0 3000.0 -s xwrd/xwrdstats -S train.scp -H xwrd/hmm8/models -M xwrd/hmm9 xwrd/treeg

//creating model in hmm10 from hmm9 based on the following command

HERest -A -D -T 1 -C config -I xwrd/xwrd.mlf -t 250.0 150.0 3000.0 -s xwrd/xwrdstats -S train.scp -H xwrd/hmm9/models -M xwrd/hmm10 xwrd/treeg

// to test the recognizer using hdecode based on the following command

HDecode -A -D -T 1 -H /home/tewodros/Desktop/htk/xwrd/hmm10/models -S /home/tewodros/Desktop/htk/testI/edited_new_test.scp -t 370.0 10000.0 -C config22 -i xwrd/recognition/xwrdrecout.mlf -w /home/tewodros/Desktop/htk/Finallm.lm -p 10.0 -s 25.0 xwrd/hdecodedictxwrd/treeg

// To generate the result using HResult by using the following command

HResults -I /home/tewodros/Desktop/htk/testI/edited_words.mlfxwrd/treegxwrd/recognition/xwrdrecout.mlf

DECLARATION

I, the undersigned, declare that this thesis is my original work and has not been presented as a partial degree requirement for a degree in any other university and that all sources used for the thesis have been duly acknowledged.

TEWODROS GIZAW

2015