ADDIS ABABA UNIVERSITY

COLLEGE OF NATURAL SCIENCE

SCHOOL OF INFORMATION SCIENCE

Spontaneous Speech Recognition for Amharic Using HMM

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE DEGREE OF MASTER OF SCIENCE IN INFORMATION SCIENCE

BY:

Adugna Deksiso

March, 2015

ADDIS ABABA UNIVERSITY

COLLEGE OF NATURAL SCIENCE

SCHOOL OF INFORMATION SCIENCE

Spontaneous Speech Recognition for Amharic Using HMM

BY:

Adugna Deksiso

March, 2015

Name and signature of members of the examining board

Name	Signature
1. _____	_____
2. _____	_____
3. _____	_____
4. _____	_____
5. _____	_____

Acknowledgments

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

Second my heartfelt thanks should go to my advisor Dr. Martha Yifiru for her constructive comments and guidance. I am thankful to her because without her guidance and genuine comments the completion of this research would have not been possible.

My special thanks go to Dr. Solomon Teferra, for his sincere clarifications and supports which helps me for this study.

I am also grateful to my friends Bantegize(Abu), Duresa and others for their support during data collection and for their comments.

Dedication

Dad, this is for you and for those who strive for love and kindness to all human beings like you.

Contents Pages

List of tables I
List of figures II
Acronyms III
Abstract IV
CHAPTER ONE 1
INTRODUCTION 1
1.1 Background 1
1.2 Statement of the Problem 7
1.3 Research Questions 8
1.4 Objective of the Study 8
1.4.1 General Objective 8
1.4.2 Specific Objectives 9
1.5 Research Methodology 9
1.5.1 Literature Review 9
1.5.2 Data Collection and Preprocessing Methods 9
1.5.3 Modeling Techniques and Tools 11
1.5.4 Testing procedure 11
1.6 Significance of the Study 11
1.7 Scope of the study 12
1.8 Organization of the Thesis 13
CHAPTER TWO 14
SPEECH RECOGNITION BASED ON STATISTICAL METHODS 14
2.1 Overview 14
2.2 Signal Processing and Feature Extraction 15
2.3 Acoustic Modeling 18
2.3.1 Hidden Markov Model (HMM) 19
2.4 Text Preparation 26
2.5 Language Model 28
2.1.1 N-gram Estimation 29
2.6 Lexical (Pronunciation) Modeling 30

2.7 Decoding (Recognizing) 31
2.8 The Hidden Markov Toolkit (HTK) 31
2.8.1 Data Preparation Tools 32
2.8.2 Training Tools 32
2.8.3 Recognition Tools 34
2.8.4 Analysis Tools 35
2.9 Spontaneous speech ASR previous works 36
CHAPTER THREE 39
AMHARIC LANGUAGE 39
3.1 Background 39
3.2 Basics of Amharic Phonetics 40
3.2.1 Articulation of Amharic Consonants 41
3.2.2 Articulation of Amharic Vowels 42
3.3 Amharic Writing System 42
CHAPTER 4 48
AMHARIC SPONTANEOUS SPEECH ASR PROTOTYPE 48
4.1 Data preparation 49
4.1.1 Pronunciation Dictionary 52
4.1.2 Transcription 53
4.1.3 Feature extraction 55
4.2 Training the model 56
4.2.1 Creating Mono-phone HMMs 56
4.2.2 Re-estimating mono-phones 58
4.2.3 Refinements and Optimization 61
4.3 Recognizer testing and evaluation 70
4.3.1 Recognizing 70
4.3.2 Analysis 72
4.4 Comparison of results and Discussion 72
4.5 Challenges 78

CHAPTER 5 79
CONCLUSION AND RECOMMENDATION 79
5.1 Conclusion 79
5.2 Recommendation 82
References 84
Appendix 89

List of tables

Table 3.1.: Categories of Amharic Consonants……41

Table 3.2., Categories of Amharic vowels……42

Table 3.3.: Number Representations in Amharic……..45

Table 3.4. Amharic fraction and Ordinal representation……..46

Table 4.1 Frequency of none speech events……68

Table 4.2 Results of cross-word and word internal tri-phones……73

Table 4.3: Results for 3 states with and without skip ……..73

Table 4.4 Analysis of results when all non-speech events modeled……74

Table 4.5 Results when most frequent non-speech events modeled……75

Table 4.6 Recognition result for speakers involved in training……77

Table 4.7 Recognition result for speakers do not involved in training……77

List of figures

Figure 1.1 Speech Processing Classifications……3

Figure 2.1 Architecture of an ASR system based on statistical approach…….15

Figure 4.1 Architecture of the system……...48

Figure 4.2 HMM model with 3 emitting state……..57

Figure 4.3 HMM model with 3 emitting state and with skip……57

Figure.4.4 Creating flat-start mono-phones……..59

Figure 4.5 Silence models……60

Figure 4.6 HMM model with 5 emitting states……..68

Figure 4.7 Summary of one time training process……69

Figure 4.8 Summary of recognition process……..71

Acronyms

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
OTH	Other Speaker
REP	Repetition
SASR	Spontaneous Automatic Speech Recognition
WER	Word Error Rate

Abstract

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

A spontaneous, speaker independent Amharic speech recognizer developed in this research work was done using conversational speeches between two or more speakers. This speech data are collected from web and transcribed manually. Among the collected data for training 2007 sentences uttered by 36 peoples from different age group and sex is used. This training data consists of 9460 unique words and it is around 3 hours and 10 minutes speech. For testing, 820 unique words which are from 104 utterances (sentences) uttered by 14 speakers are used. The collected conversational speech data contains different non-speech events both from speaker and from environment which causes the decrement of speech recognizer performance. Depending on these non-speech events frequencies, two data sets are prepared, the first data set prepared by including less frequent non-speech events in models and the second data set prepared by excluding them. Using the data sets, the acoustic model developed using word internal and cross word tied state tri-phones up to 11^{th} Gaussian mixture.

For this research, relatively the best recognizer performance is found to be 41.60% word accuracy for speakers involved in training, 39.86% for test data from both speakers which are involved and not involved in training and 23.25% for speakers those do not involved in training. The recognizer developed using cross-word tri-phone shows less performance than word internal tri-phone due to smallness of our data size. The recognizer developed and tested using the data which includes less frequent non-speech events showed less word accuracy than the one that include them.

According to the finding of this research, the performance gained for Amharic spontaneous speech recognizer is less in accuracy. This is due to the nature of speech and the smallness of the size of data used; therefore, this result can be optimized by increasing the size of the data.

CHAPTER ONE
INTRODUCTION

1.1 Background

Speech is a versatile means of communication. It conveys linguistic (e.g., message and language), speaker (e.g., emotional, regional, and physiological characteristics of the vocal apparatus), and environmental (e.g., where the speech was produced and transmitted) information. Even though such information is encoded in a complex form, humans can relatively decode most of it [1].

This human ability has inspired researchers to develop systems that would imitate such ability. Different researchers have been working on several fronts to decode most of the information from the speech signal. Some of these fronts include tasks like identifying speakers by the voice, detecting the language being spoken, transcribing speech, translating speech, and understanding speech. Among all speech tasks, automatic speech recognition (ASR) has been the focus of many researchers for several decades. In this task, the linguistic message is one of the areas of interest [2].

Automatic speech recognition sometimes referred to as just speech recognition, computer speech recognition (erroneously as voice recognition) is the process of converting speech signals uttered by speakers into a sequence of words, which they are intended to represent, by means of an algorithm implemented as a computer program. The recognized words can be the final results, as for applications such as data entry and dictation systems or the words so recognized can be used to trigger specific tasks as in command and control applications [1].

Automatic Speech Recognition Types

Speech recognition systems can be categorized based on different parameters, some of the parameters and types of Automatic speech recognizers depending on these parameters are given below [2]:

Based on Speaking Mode: Isolated (discrete) and continuous speech

Isolated (Discrete) speech recognition systems are systems that require the speaker to pause briefly between words. As it is explained by Markowitz [3], Speech is said to be continuous when it is uttered as a continuous flows of sounds with no inherent separations between them and speech recognition system developed using these type of speech is referred to as continuous speech recognition.

Based on Enrollment: Speaker-dependent and speaker-independent

Speaker-dependent system uses speech samples from the target speaker to learn the model parameters of the speaker's voice. Speaker independent systems are designed to be used by any users who want to use them with no enrollment. This is also planned to be used for this study.

Based on Vocabulary size: small, medium and large

Small vocabulary speech recognition has word size of 1 to 1,000 words, medium corpus speech recognition contains from 1,000 to 10,000 words and large corpus has more than 10,000 words.

Based on Speaking Style: Read speech and Spontaneous speech

Read speech is a speech which is made ready by the form of script and reader inserts false pauses between words while reading the text. If compared with spontaneous speech these speeches are more fluent and have less non speech events like filled pause, repetitions, hesitation and others. We can use these speech data for the development of speech recognizer; therefore we can say it is read speech recognizer [4].

Spontaneous speech is conversational and it is not well structured, acoustically and syntactically, as read speech. The presence of dis-fluencies makes the spontaneous speech disparate and provides a challenge for speech processing. State-of-the-art automatic speech recognition has achieved high recognition accuracy for read speech [5]. However, the accuracy is still poor for spontaneous speech with dis-fluencies.

Among the ASR types briefly described above, with this study we have developed continuous spontaneous speech recognition which is speaker independent using medium vocabulary size.

The summary of speech processing and their classification are given below briefly in figure 1.1.

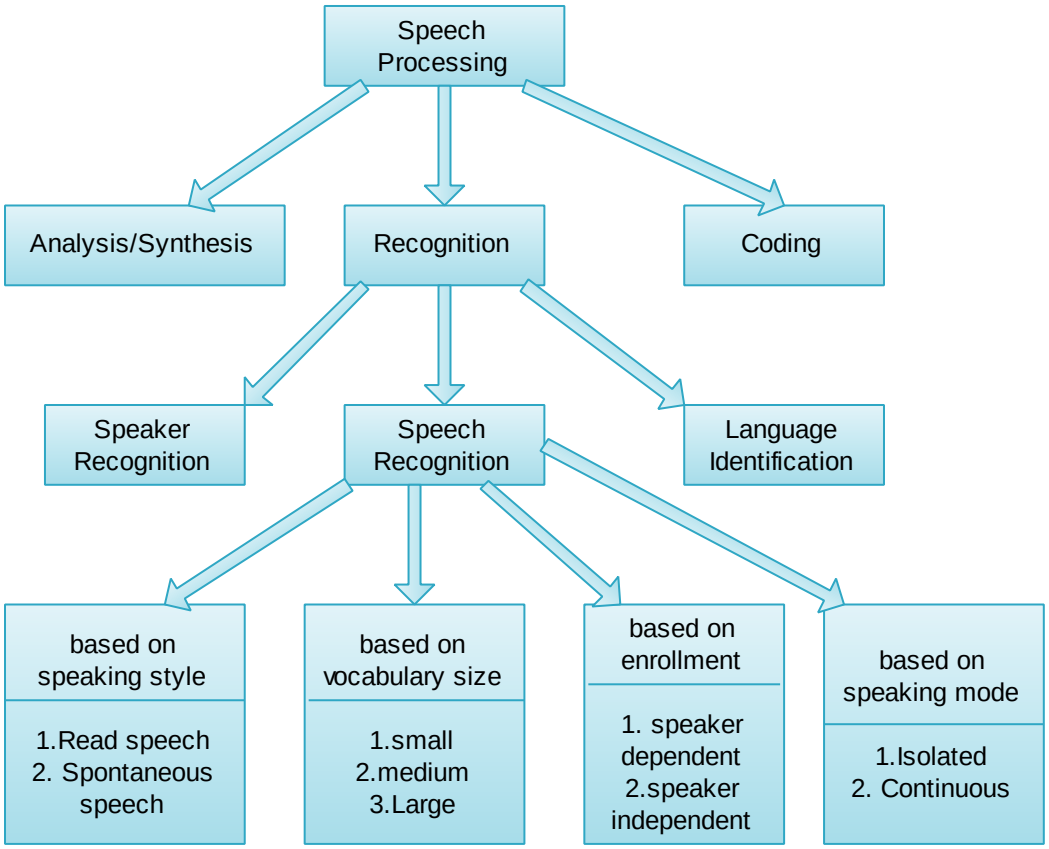

Figure 1.1 Speech Processing Classifications, adapted from [2]

Automatic Speech Recognition Components

There are three important models which are needed for the recognition and they are important components of speech recognition systems. They are Acoustic model, lexical model (pronunciation dictionary) and language model. These components work together in speech recognition system [6].

The acoustic model provides the probability that when the speaker utters, a word sequence the acoustic processor produces the representation of the word sequence.

The pronunciation Dictionary (lexical model) is a language dictionary which contains mapping of each word to a sequence of sound units. The purpose of this file is to derive the sequence of sound units associated with each signal. 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. Pronunciation dictionary which is used for this study is canonical.

Units of recognition

The most popular units of speech for speech recognition development are sub-word units (such as context independent phone, context dependent phones and syllables) and Words. For the better performance of speech recognizer the unit of speech which is preferred for speech recognition development should be trainable, well defined and relatively insensitive to context.

Phone is trainable since there are few phones in any language. But phones are more sensitive to context and they do not model co-articulation effects. These demerits of phones decrease the performance of recognizer. In order to overcome these drawbacks Rabiner and Juang [7] suggests that other speech units can be considered for speech recognition modeling. Word-dependent tri-phones and context-dependent phones or tri-phones take context in to consideration. Word-dependent models can model context than phones, but they require large training data and storage. Tri-phone models are phone models that take left and right neighboring phones into consideration [8]. Although they are many in number and they consume much memory, tri-phone modeling is powerful since it models co-articulation and insensitive to context than phone modeling. These both units of recognitions are used for this study and their results compared.

The language model means providing the behavior of the language. The language model describes the likelihood or the probability taken when a sequence or collection of words is seen. A language model is a probability distribution over the entire sentences/texts. The purpose of creating a language model is to narrow down the search space, constrain search and thereby to significantly improve recognition accuracy.

Automatic Speech Recognition Approaches

Automatic speech recognition is the independent, computer-driven transcription of spoken language into readable text in real time. To do this the features of the speech should be extracted and they have to be modeled. To model the distribution of the feature vectors different modeling techniques can be used depending on the recognition approach used. Jurafsky et.al [1] states that, most of the times there are four basic speech recognition approaches:

I. Rule-Based (Acoustic-phonetic) approach
II. Template-Based approach
III. Stochastic (Statistical) approach
IV. Artificial Intelligence approach

I. Acoustic-phonetic Approach

Acoustic-phonetic also called rule-based approach uses knowledge of phonetics and linguistics to guide search process. Usually some rules are defined expressing everything (anything) that might help to decode: Phonetics, phonology, Syntax and Pragmatics.

In the Acoustic Phonetic approach the speech recognition are based on finding speech sounds and providing appropriate labels to these sounds . This is the basis of the acoustic phonetic approach which postulates that there exist finite, distinctive phonetic units (phonemes) in spoken language and that these units are broadly characterized by a set of acoustics properties that are manifested in the speech signal over time.

This approach can perform Poor due to:

- Difficulty to express rules
- Difficulty to make rules interact
- Difficulty to know how to improve the system

II. Template Based Approach

Template-based approach Store examples of units (words, phonemes, syllables), then find the example that most closely fits the input. It extracts features from speech signal, and then it matches these which have similar features.

The drawbacks of this approach are:

- ✓ It works for discrete utterances and for a single user.
- ✓ Hard to distinguish very similar templates.
- ✓ The performance quickly degrades when input differs from templates.

III. Stochastic (Statistical) Approach

This approach is an extension of template-based approach, using more powerful mathematical and statistical tools. Sometimes it is seen as anti-linguistic approach. Statistical approach uses the probabilistic models to deal with uncertain and incomplete information found in speech recognition the most widely used model is HMM. This Approach works by collecting a large corpus of transcribed speech recordings then Train the computer and then at run time, apply statistical processes to search through the space of all possible solutions, and pick the statistically most likely one.

The statistical approach, involves two essential steps namely, pattern training and pattern comparison. This approach is widely implemented for ASR developments using different modeling methods. Among these methods HMM is the most popular one and we have used for this study also. We have used this statistical pattern recognition approach, since it has different advantages over the other three approaches. The essential feature of this approach is that it uses a well formulated mathematical framework and establishes consistent speech pattern representations for reliable pattern comparison from a set of labeled training samples via a formal training algorithm [1].

IV. Artificial Intelligence Approach

The main idea of this approach is collecting and employing the knowledge from different sources in order to perform recognition process. The knowledge sources contain acoustic, lexical, syntactic, semantic and pragmatic knowledge which are important for speech recognition system.

The Artificial Intelligence approach is a hybrid of the acoustic phonetic approach and pattern recognition approach. In this, it exploits the ideas and concepts of Acoustic Phonetic and Pattern Recognition methods. Knowledge based approach uses the information regarding linguistic, phonetic and spectrogram [9].

1.2 Statement of the Problem

Previous attempts to build automatic Amharic speech recognizers are very limited in number. Solomon [10] Built both speaker dependent and independent, isolated syllable recognizers. Kinfe [11] Conducted study on sub-word based Amharic speech recognizer. Martha [12] developed a small vocabulary, isolated word recognizer for command and control interface to Microsoft Word. Zegaye [13] developed a speaker independent, continuous Amharic speech recognizer. Solomon [6] developed a syllable-based, large vocabulary, speaker independent, continuous Amharic speech recognizer. Yitagesu [14] demonstrated a new approach that, a smaller number of acoustic models are sufficient to build a syllable based, speaker independent, continuous, Amharic ASR. All of the described researches have done using HMM.

Hussien [15] Tried a different approach by mixing artificial neural networks and HMM to build a speaker independent continuous speech recognizer for Amharic. Yitagesu [14] has demonstrated that a smaller number of acoustic models (only for 93 syllables) are sufficient to build a syllable based, speaker independent, continuous, Amharic ASR. They built for weather forecast and business report applications using the UASR (Unified Approach to Speech Synthesis and Recognition) Tool kit.

The growing demand for reliable spontaneous speech recognizers has been exhibited in applications such as dialogue systems, spoken document retrieval, call managers and automatic transcription of lectures and meetings.

The previous attempts on Amharic ASR done using read speech data and domain based spontaneous speech for dictation. To our knowledge ASR using general domain Amharic spontaneous speech data is not developed yet that is why we have developed in this study.

> *"The ultimate aim of research in speech technology is the development of human-computer conversational system that communicates with any one, about anything, on any topic and in any situation." [16]*

Therefore the aim of this study is to develop a recognizer which is speaker independent that can be used in different domain and different environment. Since we considered that it is a good input for this ultimate aim, we have tried our best to develop a recognizer which is speaker independent using spontaneous speech from different domain.

1.3 Research Questions

The study tried to answer the following research questions.

- ✓ What are the challenges of Amharic spontaneous speech recognition system development?
- ✓ What are the effects of sentence length on the performance of Amharic spontaneous speech recognizer?
- ✓ What are the effects of modeling non-speech events on speech recognizer performance?

1.4 Objective of the Study

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

1.4.1 General Objective

The general objective of this study is to explore the possibility of developing Amharic spontaneous speech recognition system using HMM.

1.4.2 Specific Objectives

Specific objectives of the research are:-

- ✓ To develop spontaneous speech corpus that can be used for training and testing purpose.
- ✓ To identify feature of spontaneous speech.
- ✓ To build a prototype speaker-independent medium vocabulary spontaneous speech recognizer using Hidden Markov Model (HMM).
- ✓ To test the performance of the developed recognizer prototype using test corpus.
- ✓ To analyze the results and give conclusion and forward recommendations.

1.5 Research Methodology

The following methods were used in conducting this study.

1.5.1 Literature Review

Exhaustive literature review was performed to investigate the underlying principles/theories of the various approaches, techniques and tools that were employed in the research. Literatures on the Amharic language and on tools and models implemented for this study were reviewed. To be informed what others have done in this area and to better understand the problem, a comprehensive review of available literatures on automatic speech recognition was conducted.

1.5.2 Data Collection and Preprocessing Methods

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. These audio and text data are applied according to their importance for where they are appropriate.

Speech Data

The audio data which is used in this study are collected from different online multimedia sources like YouTube and DireTube. These audio files are with 44100 Hz sampling rate recorded by different local mass media particularly from Sheger 102.1 FM radio, Ethiopian Broadcasting Corporate (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, these speeches are segmented in to sentences and transcribed manually. Even if it was one of the challenges we face, we have tried to ignore from our corpus the sentences with some foreign words during our audio collection.

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 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.

Text Data

Just listening and writing these segmented audio in to their equivalent text was the most challenging and time consuming task in data preparation process.

The speeches equivalent orthographies (texts) of audio files are also used for pronunciation dictionary development (lexical modeling) and for language modeling. The language model which is used for this study is developed using the texts transcribed from audio files we have used and the texts obtained from Solomon [6]. The texts we have taken from him are in Unicode format, since our tool does not support this encoding we have transliterated the texts in to its equivalent ASCII format using python code we have prepared for this purpose. After format conversion both the texts from Solomon [6] and our texts are used for development of language models and implemented for where they required.

The recognition unit for this speech recognition is sub-word unit particularly phones, tri-phones (context dependent and cross-word tri-phones). The vocabulary (words) used for training in this experiment, excluding sp, sil and phones assigned for non-speech, it consists of 36 Amharic phones out of 38 total phones.

1.5.3 Modeling Techniques and Tools

For the development of speech recognizer the selection of modeling tools is the most important step of the process. We have used Hidden Markov Model modeling technique that became the predominant technique for speech recognition. HMMs are at the heart of almost all modern speech recognition systems especially the system which is used statistical method, although the basic framework has not been changed significantly in the last decade or more.

For this study, HTK (Hidden Markov Model toolkit) has been employed. This toolkit was preferred since different studies in this area had used the toolkit and achieved considerable results. In addition to this, this toolkit is freely available for academic and research use. For language modeling we have used SRILM language modeling toolkit and for text normalization and preparation we have also used Python and Perl codes. The audio file is segmented into sentences using PRAAT tool. Notepad++, visual studio and other software are used for text editing and for purposes where they needed.

1.5.4 Testing procedure

The testing is done using test data prepared for this purpose, after development of acoustic model as a result of training, lexical models (pronunciation dictionary) and language models. For testing we have implemented HTK modules HVite and HDecode which works with word internal tri-phones and cross word tri-phones respectively. Then by taking the recognized output label file, HTK module HResults is used for performance analysis of developed recognizer.

1.6 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 conversational (spontaneous) speech therefore peoples need

to communicate with machine by conversational speech like they do with people; this study serves as one attribute to answer this interests for Amharic speakers. Therefore the result of this study can also be used as an input towards the development of human computer conversational system.

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 speech recognition system since they have difficulty in using keyboard and mouse to write commands and control computers. Other group of users that can get benefit from speech recognition system is people whose eyes and hands are busy in performing other task. In general, it can be said that if well done and ready for application, this system is helpful for any people who can speak Amharic since it is speaker independent and also it is general domain. This study is, therefore, a step towards the development of such a useful system.

There were some attempts of studying ASR using read speech data, but this research is done using conversational speech data. Therefore, this study has its own contribution on the applicability of Amharic speech recognition, since effectively broadening the application of speech recognition depends crucially on raising recognition performance for spontaneous speech. The ultimate goals of ASR studies are speaker-independent continuous speech recognition system. Since this study conducted on speaker-independent and conversational speech it will have its own significance for the ultimate goal of ASR.

This study can be used as an input for future researches on Amharic speech recognition since there are recommendations from this study finding for future works in this area, particularly in spontaneous speech recognition.

1.7 Scope of the study

This study is held on spontaneous speech recognition for Amharic language. It is speaker independent and uses small corpus of speech which is prepared using conversational speech data collected from web.

Stochastic approach is used with the well-established model which is HMM model; it is neither with neural networks nor hybrid models. Language model we have developed for this experiment was done using small data in size and it is bigram.

The pronunciation dictionary used for training and testing was canonical pronunciation dictionary prepared by taking phones 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.

1.8 Organization of the Thesis

This paper is divided into 5 chapters. Chapter one consists of background, statement of the problem, research question, objectives of the study, methodology followed in the course of the study and the scope the study. In chapter two statistical methods based speech recognition is reviewed. Chapter three presents Amharic language. Chapter four provides the development prototype of Amharic spontaneous ASR system. Finally, conclusions and recommendations are given in chapter five.

CHAPTER TWO

SPEECH RECOGNITION BASED ON STATISTICAL METHODS

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 [17].

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 $W = (w_1,w_2,w_3 \ldots w_i)$, 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 [7].

The main components of a generic statistical speech recognition system are show in Figure 2.1 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.

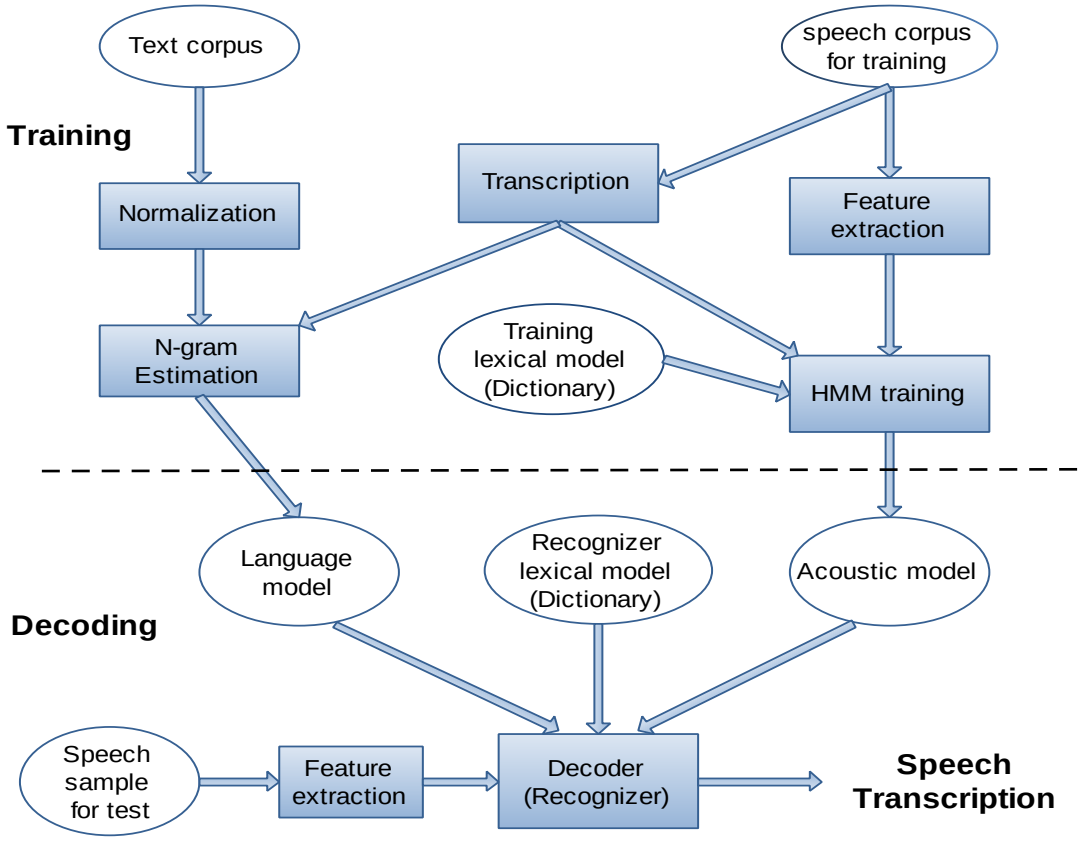

Figure 2.1 Architecture of an ASR system based on statistical approach, adapted from [18]

2.2 Signal Processing and Feature Extraction

Hermansky [19] Indicated that, every other component in a speech recognition system depends on two basic subsystems: signal processing and feature extraction. 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.

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.

There are different methods that can be used to extract parameters of a speech, **Signal based,** method which describe the signal in terms of its fundamental components, **production-based and perception based** that works by simulating the effect that the speech signal has on the speech perception system [19].

Signal based Analysis

The methods in this type of analysis disregard how the speech was produced or perceived. The only assumption is that the signal is stationary. Two methods commonly used are **filter banks** and **wavelet transforms** [19].

Filter banks estimate the frequency content of a signal using a bank of band pass filters, whose coverage spans the frequency range of interest in the signal (e.g., 100-3000Hz for telephone speech signals, 100-8000 Hz for broadband signals). The most common technique for implementing a filter bank is the short-time Fourier transform (STFT). It uses a series of harmonically related basis functions to describe a signal.

The drawbacks of the STFT are that all filters have the same shape, the center frequencies of the filters are evenly spaced and the properties of the function limit the resolution of the analysis [19]. Another drawback is the time-frequency resolution trade-off. A wide window produces better frequency resolution (frequency components close together can be separated) but poor time resolution. A narrower window gives good time resolution (the time at which frequencies change) but poor frequency resolution.

Given the STFT-based filter bank drawbacks, wavelets were introduced to allow signal analysis with different levels of resolution. This method uses sliding analysis window function that can dilate or contract, and that enables the details of the signal to be resolved depending on its temporal properties. This allows analyzing signals with discontinuities and sharp spikes [9].

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. Hermansky [19] states 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. 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.

Even though this model is a decent approximation of the speech production, it fails on explaining the production of voiced fricatives. Voiced fricatives are produced using a mix of excitation sources: a periodic component and an aspirated component. Such mix of sources is not taken into account by the source-filter model.

Several methods take advantage of the described linear model to derive the state of the speech production system by estimating the shape of the filter function. There are three most popular production-based analyses*: spectral envelope, linear predictive analysis and cepstral analysis* [19].

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) [20].

Mel-Frequency Cepstrum Coefficients (MFCC)

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 [20]:

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

For frequency f, the scale is close to linear for frequencies below 1 kHz and is close to logarithmic for frequencies above 1 kHz [20]. MFCCs which are implemented for this study are often used in many other speech recognition systems.

2.3 Acoustic Modeling

After some preprocessing (for instance, speech signal processing and feature extraction) it is possible, to represent the speech signal as a sequence of observation symbols $\mathbf{O} = o_1\, o_2 \ldots o_T$ that represents a string composed of elements of a particular alphabet of symbols. Then mathematically the speech recognition problem comes down to finding the word sequence **W** having the highest probability of being spoken, given the acoustic evidence **O**, thus we have to solve: [21]

$$\mathbf{W} = \arg max\, (P(\mathbf{W} \mid \mathbf{O}) \qquad \text{2.2}$$

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 as described by Wigger, et.al [21] Bayes formula gives:

$$P(\mathbf{W} \mid \mathbf{O}) = (P(\mathbf{W})P(\mathbf{O} \mid \mathbf{W}))/(P\,(\mathbf{O})\,) \qquad \text{2.3}$$

From the above formula, $P(\mathbf{W})$, is called the language model, which is the probability that the word string **W** will be uttered and $P(\mathbf{O}|\mathbf{W})$ 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 $P(\mathbf{O})$ 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:

$$\mathbf{W} = \arg max\, P(\mathbf{W})P(\mathbf{O} \mid \mathbf{W}) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 2.4$$

The acoustic model determines what sounds will be produced when a given string of words is uttered. Thus for all possible combinations of word strings **W** and observation sequences **O** the probability P (**O**| **W**) must be available. This number of combinations is just too large to permit a lookup; in the case of continuous speech it's even infinite. It follows that these probabilities must be computed on the fly, so a statistical acoustic model of the speakers' interaction with the recognizer is needed.

The most frequently used acoustic model these days is the Hidden Markov model [21], which is also implemented for this study.

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 [1].

As described by Jurafsky, et.al [1] 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 set

Q:

A Hidden Markov model can be defined by the following parameters:

- $S = \{s_1, s_2, ... s_N\}$: 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 $\mathbf{s_t}$, thus $s_t = i$ means that the model is in state ***i*** at time ***t.***
- $A = a_{11}\ a_{12}\ ...\ a_{ij}$: A transition probability A, each a_{ij} representing the probability of moving from state i to state j.
- $O = o_1\ o_{2\ ...}\ o_N$: A set of observations, each one drawn from a vocabulary $V = v_1, v_{2...}v_v$
- 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.
- $\pi = \pi_1, \pi_{2, ...}, \pi_N$: An initial probability distribution over states :$\boldsymbol{\pi_i}$ is the probability that $\boldsymbol{s_i}$ 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 $\lambda = (A, B, \pi)$, 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 [21].

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

$$\alpha_t(j)= P(o_1,o_2,\ldots o_t,q_t = s_i| \lambda) \quad \ldots\ldots 2.5$$

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

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

The three factors that are multiplied equation 2.6 for extending the previous paths to compute the Viterbi probability at time t are:

$\boldsymbol{\alpha_{t-1}(i)}$ The **previous forward path probability** from the previous time step

$\boldsymbol{\alpha_{ij}}$ The **transition probability** from previous state $\mathbf{q_i}$ to current state $\mathbf{q_j}$

$\mathbf{b_j(o_t)}$ The **state observation likelihood** of the observation symbol $\mathbf{o_t}$ given the current state **j**

We can define the forward algorithm using a statement of the definitional recursion:

- ✓ Initialize

$$\alpha_1(i) = \pi_i b_i(o_1) 1 \le i \le N \quad \ldots\ldots 2.7$$

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

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

- ✓ Termination

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

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 [7]. 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 $Q=q_1 \ldots q_T$, such that:

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

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

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

The three factors that are multiplied equation 2.11for extending the previous paths to compute the Viterbi probability at time t are:

δ_{t-1} (i) the **previous Viterbi path probability** from the previous time step

a_{ij} the **transition probability** from previous state $\mathbf{q_i}$ to current state $\mathbf{q_j}$

$b_j(o_t)$ the **state observation likelihood** of the observation symbol $\mathbf{o_t}$ given the current state **j**

A formal definition of the Viterbi recursion can be as follows:

1. Initialize

$$\delta_1(i) = \pi_i b_j(o_1) \qquad 1 \le i \le N \quad \text{.....2.12}$$

2. Recursion

$$\delta_t(j) = \max_{1 \le i \le N} (\delta_{t-1}(i) a_{ij}) b_j(o_t) \quad \text{.....2.13}$$

$$\psi_t(j) = \left[\arg\max_{1 \le i \le N} \delta_{t-1}(i) a_{ij} \right] \quad 2 \le t \le T, 1 \le j \le N \quad \text{.....2.14}$$

3. Terminate

$$p^* = \max_{1 \le i \le N} \delta_T(i) \quad \text{.....2.15}$$

P* gives the state-optimised probability

$$q_T^* = \arg\max_{1 \le i \le N} \delta_T(i) \quad \text{.....2.16}$$

Q* is the optimal state sequence; Q*={q_1*, q_2* … q_T*}

4. Backtrack state sequence

$$q_t^* = \psi_{t+1}(q_{t+1}^*) \qquad t = T-1,...,1 \quad \text{.....2.17}$$

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 [7] 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:

i. Initial state distribution:π_i
ii. Transition probabilities: $a_{i,j}$
iii. Emission probabilities: $b_i(o_t)$

i. Re-estimating the transition probabilities

Here we have to solve, 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:

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

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

Let $\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j)$ is the probability of being in state s_i, given the complete observation O.

the above equation can be modified as:

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

ii. Re-estimating Initial state probability

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

Re-estimation is: $\hat{\pi}_i = \text{expected number of times in state } s_i \text{ at time } 1$

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

iii. Re-estimation of Emission probabilities

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

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

Where $\delta(o_t, v_k) = 1, \text{if } o_t = v_k, \text{and 0 otherwise}$

Finally After Baum welch algorithm implementation we updated our model from $\lambda = (A, B, \pi)$, to $\lambda' = (\hat{A}, \hat{B}, \hat{\pi})$ by re-estimating the above three probabilities.

2.4 Text Preparation

Variety of text materials that are useful can be usually used for language model, lexical model, and for training purposes. These text data can be obtained from transcription of speech data and other sources.

Spontaneous speech, in addition to normal speeches it contains some non-speech events. These non-Speech events can be Filled pauses, lip smack, breathing, long pause, repetitions, hesitations and many others. In the process of speech recognizer development since they have effects, not only normal speeches but also these non-speech events should be modeled [23]. So like normal speeches we have to transcribe them.

Generally there are different types of non-speech events but in this paper we have transcribed and modeled only some of them which are observed in our speech data. Filled pause (FP), hesitation (HES), breathe (BR), Other Speaker (OTH), Throat clear (THC), Laugh (LGH), Lip smack (LIP), repetition and Interruption (INT) of other speaker. Non speech events like background noise and breathe at the end and start of the sentence are not modeled since they can be tolerated by the tool used.

Depending on their sources, these non-speech events that occur in speeches can be from one of below given three major sources [23]. (1) Speaker Generated (2) Other Speaker Distortions (3) Background Noise.

Speaker-generated non-speech events

Speaker generated non-speech events are Hesitation (HES), Lip smack (LIP), Repetition, Filled Pause (FP), Throat Clear (THC), Breathe (BR) and Laugh (LGH). Since they are produced by the speaker the same way as speech, Speaker-generated non-speech events always occur between words and sometimes in a rare case with words like in the case of speaking while laughing. They can be therefore annotated as another word, using a symbols assigned for each of them (e.g. if there is filled pause between two words, ‘word1 FP word2’).

FP መጀመሪያ FP ሚኒሊክ ትምህርት ቤት ነው የሰራሁት FP መነባነብ ነው

[FP mäyämärija FP miniliki timihiriti beti näwi yäsärahuti FP mänäbanäb inäwi]

Repetitions of the same words are also common challenges in the natural informal speech which is not read. The repeated words are sometimes contextually important but sometimes the speakers' simply repeat the word (words) which do not have any change on the meaning. Sometimes they may repeat partial part of the word in order to correct it, in this case it is transcribed as a hesitation. But at the time of complete repetition of the word which does not bring any change on the meaning of the sentence, it is identified by '~' symbol. But the pronunciation dictionary of the word is assigned as a normal word.

እሱ~እሱ እንዴት መጣና አየሽ ደራሲው [?isu~?isu ?inidetimäTana ?ajäSidärasiwi]

ከዛን ጊዜ ከዛንጊዜ ጀምሮ በቃ ያው አቆምኩት [käzanigize ~käzani ~gize dʒämiro bäqa jawi ?aqomikuti]

Due to the naturalness of spontaneous speeches the irregular lengths of words are exist.

Other speaker in background

As the speech is from conversation between two and more person in different environments, the audience present in the room can influence the speech and distortion from other speaker can be present. In the case of no audiences in the room the interviewee and the interviewer can speak simultaneously. We can resolve two different situations here, either the distortion appear within speech pause (OTH) or more speakers are talking simultaneously like in case of cocktail. The distortion appears within speech pause particularly between words can be transcribed:

ቀበድ አለው ወይ OTH እያልኩ አሾፍበታለው[qäbidi ?aläwiwäji OTH ?iyaliku ?aSofibätaläwi]

Background Noise events

Environmental distortion can overlap particular words, so we need to use special rules for better description of noise disturbance within the speech. If the noise appears only in the pause between words, it is marked similarly to speaker-generated events. But if the following speech is affected, starting (INT-) and ending (-INT) mark is used like, e.g. "word1 INT− word2 –INT word3 INT-word4-INT".

ከህዝብ ጋር በደንብ ተገናኘሁ ማለት ነው[INT-kähizibi-INT gari bädänibi tägänaɲähu maläti näwi]

Given a large text corpus it may seem relatively straightforward to construct n-gram language models. Most of the steps are pretty standard and make use of tools that count word sequence occurrences. The main considerations are the choice of the vocabulary, the definition of words (treatment of compound words and acronyms), and the choice of the LM back-off strategy. There is, however, a significant amount of effort needed to process (or normalize) the texts before they can be used. One motivation for the normalization is to reduce lexical variability so as to increase the coverage for a fixed size task vocabulary. The processing decisions are generally language-specific [24].

2.5 Language Model

Language models (LMs) capture regularities in spoken language and are used in speech recognition to estimate the probability of word sequences. While grammatical constraints described by hand-crafted context-free grammars have been used for small to medium size vocabulary tasks, large vocabulary continuous speech recognition(LVCSR) is essentially always based on data driven approaches. The most popular statistical method is the so called n-gram model, while attempts to capture the syntactic and semantic constraints of the language by estimating the frequencies of sequences of **n** words [24]. The assumption is made that the probability of a given word string W = (ω_1,ω_2,…,ω_k)can be approximated by the following forward sequential decomposition

$$P(W) = \prod_{i=1}^{K} \Pr(\omega_i | \omega i - n + 1, \dots, \omega i - 2, \omega i - 1) \quad \text{.....2.24}$$

There by reducing the word history to the preceding $n-1$ word. It should be noted that other decomposition of P(W) can also be appropriate, for example, a backward decomposition will lead to a backward n-gram model.

A pre-requisite for estimating n-gram language models is the availability of appropriately processed text corpora. Language models are usually estimated from manual transcriptions of speech corpora and from normalized text corpora. To ensure accurate models, the texts need to be as representative as possible of the expected audio input to be transcribed. Text preparation entails locating appropriate sources of text data and audio transcriptions, and processing them in a homogeneous manner.

2.1.1 N-gram Estimation

Using the maximum likelihood (ML) criterion, the n-gram probabilities are estimated from the frequencies of the word sequences of length N in the training corpus (texts or speech transcriptions).A statistical language model assigns a probability to a sequence of *i* words $P(w_1,w_2,w_3 \ldots w_i)$ by means of a probability distribution. Using HMM by considering different number of histories N we can develop many, statistical n-gram language models. Unigram, Bigram and Trigram are the most popular ones but it is possible to go beyond this by increasing the number of histories used to estimate the probability [6]. Of course it is possible to create any *N*-gram, which is *N*> 3, but it causes a combinatorial increase in model complexity and size.

The **Bigram language** model approach uses only one step histories.

$$P(\mathbf{W}) = P(w_1)P(w_2 \mid w_1) \ldots P(w_i \mid w_{i-1}) \quad \ldots\ldots 2.25$$

It is now easy to estimate the probabilities $P(w_i|w_{i-1})$ simply by counting the number of times each word pair occurs in a representative corpus of strings. One problem with this procedure is that the training corpus might be missing some words, which do occur in the vocabulary the recognizer uses. To these words an estimated probability of zero would be assigned. Therefore, usually a small portion of the probability distribution is reserved for words out of the vocabulary that do not occur in the training corpus.

Most present-day speech recognizers take it one step further and use the **trigram language model** that has two word histories. This language model is powerful but requires large amounts of training data to provide values for all probabilities $P(w_i \mid w_{i-1}, w_{i-2})$. Furthermore the model tends to get very large as all the words have to be listed and for each word all possible two-word histories must be provided. To avoid these problems some extensions to the basic statistical language models are possible.

To make up for data sparseness the trigram frequencies can be smoothed by interpolating trigram, bigram and unigram relative frequencies.

$$P(wi \mid w_{i-1}, w_{i-2}) = \lambda_1 f(w_i) + \lambda_2 f(w_i \mid w_{i-1}) + \lambda_3 f(w_i \mid w_{i-1}, w_{i-2}) \quad \ldots\ldots\ldots\ldots\ldots 2.26$$

Where the weights satisfy $\lambda 1+\lambda 2+\lambda 3 = 1$.

Backing-off is another method to solve the problem for the lack of training data used in many speech recognizers. It considers, if there is enough evidence the trigram frequency is a good estimate for the probability, if not the system should back-off and rely on bigrams and if there is not enough evidence even for those, unigrams should be used.

2.6 Lexical (Pronunciation) Modeling

The input speech is divided into small components like phones, tri-phones, syllables. The pronunciation dictionary (lexical model) determines how these small components come together and gives word. It is the task of pronunciation dictionary to construct (transcribe) correct word by concatenating these small components of speech (phones, syllable, or other). There are two types of pronunciation dictionary, namely canonical and alternative.

The pronunciation dictionary is the link between the acoustic-level representation and the lexical items output by the speech recognizer. The accuracy of the acoustic models is partly dependent upon the consistency of the pronunciation dictionary. Associated with each lexical entry are one or more pronunciations, described using the chosen elementary units (usually phonemes or phones). This set of units is evidently language dependent. For example, some commonly used phone set sizes are 45 for English, 50 for German and Italian, 35 for French and Mandarin(to which tones may be added), and 25 for Spanish.

In generating pronunciation base forms, most lexicons include standard full-form pronunciations and do not explicitly represent phonetic variants. More importantly, there often is a continuum between different phonetic realizations of a given phoneme and the decision as to which occurred in any given utterance is subjective. By using a phone representation, no hard decision is imposed, and it is left to the acoustic models to represent the observed variants in the training data.

2.7 Decoding (Recognizing)

The Statistical Speech Recognition decoding is the design of an efficient search algorithm to deal with the huge search space obtained by combining the acoustic and language models. The aim of the decoder is to determine the most likely word sequence W, given the language model, the pronunciation dictionary and the acoustic models, Recognition (Pattern matching) refers to the process of assessing the similarity between two speech patterns, one of which represents the unknown speech and one of which represents the reference pattern (derived from the training process) of each element that can be recognized.

The reference pattern consists of a probabilistic model, such as an HMM, and the process of pattern matching is done using the statistical knowledge contained in the probabilistic model to assess the likelihood of the speech (which led to the model) being realized as the unknown pattern.

2.8 The Hidden Markov Toolkit (HTK)

There are several toolkits available that implement the algorithms needed in speech recognition, allowing a recognizer designer to focus on the important tasks involved in building a speech recognizer like data preparation, training and recognizers evaluation. One of these toolkits is the Hidden Markov Toolkit (HTK). The discussion about this tool will be continued below, as it is referred from Young, et.al [25], The HTK Book for HTK version 3.4.

HTK is a portable software toolkit for building and manipulating systems that use continuous density Hidden Markov models. It has been developed by the Speech Group at Cambridge University Engineering Department. HMMs can be used to model any time series and the core of HTK is similarly general purpose. However, HTK is primarily designed for building HMM based speech processing tools, in particular speech recognizers. It can be used to perform a wide range of tasks in this domain including isolated or connected speech recognition using models based on whole word or sub- word units, but it is especially suitable for performing large vocabulary continuous speech recognition.

HTK includes nineteen tools that perform tasks like manipulation of transcriptions, coding data,

various styles of HMM training including Baum-Welch re-estimation, Viterbi decoding, results analysis and extensive editing of HMM definitions.

2.8.1 Data Preparation Tools

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

Although all HTK tools can parameterize waveforms on-the-fly, in practice it is usually better to parameterize the data just once. 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 the 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. HLEd is a script-driven label editor which is designed to make transformations to label files, like translating word level label files to phone level label files, merging labels or creating tri-phone labels. HLEd can also output files to a single Master Label File MLF, which is usually more convenient for subsequent processing. HLStats can gather and display statistics on label files and where required, HQuant can be used to build a VQ codebook in preparation for building discrete probability HMM system.

2.8.2 Training Tools

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. 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.

If segmented transcriptions are available the tools HInit and HRest provide isolated word style training using the fully labeled data as bootstrap data. HInit can be used to provide initial estimates of whole word models in which case the observation sequences are realizations of the corresponding vocabulary word. Alternatively, HInit can be used to generate initial estimates of seed HMMs for sub-unit based speech recognition. In this latter case, the observation sequences will consist of segments of continuously spoken training material. HInit will cut these out of the training data automatically by simply giving it a segment label. In both of the above applications, HInit normally takes as input a prototype HMM definition, which defines the required HMM topology i.e. it has the form of the required HMM except that means, variances and mixture weights are ignored. The transition matrix of the prototype specifies both the allowed transitions and their initial probabilities. Transitions, which are assigned zero probability, will remain zero and hence denote non-allowed transitions. HInit estimates transition probabilities by counting the number of times each state is visited during the alignment process.

HRest performs basic Baum-Welch re-estimation of the parameters of a single HMM using a set of observation sequences. HRest can be used for normal isolated word training in which the observation sequences are realizations of the corresponding vocabulary word or it can be used for isolated model training for sub-unit based speech recognition. In this latter case, the observation sequences will consist of segments of continuously spoken training material. HRest will cut these out of the training data automatically by simply giving it a segment label. In both of the above applications, HRest is intended to operate on HMMs with initial parameter values estimated by HInit.

HERest is used to perform a single re-estimation of the parameters of a set of HMMs using an

embedded training version of the Baum-Welch algorithm. Training data consists of one or more utterances each of which has a transcription in the form of a standard label file (segment boundaries are ignored). For each training utterance, a composite model is effectively synthesized by concatenating the phoneme models given by the transcription. Each phone model has the same set of accumulators allocated to it as are used in HRest but in HERest they are updated simultaneously by performing a standard Baum-Welch pass over each training utterance using the composite model.

The tool HHEd is a HMM definition editor which will clone models into context-dependent sets, apply a variety of parameter tying and increment the number of mixture components in specified distributions. To improve performance for specific speakers the tools HEAdapt and HVite can be used to adapt HMMs to better model the characteristics of particular speakers using a small amount of training or adaptation data.

2.8.3 Recognition Tools

HTK provides a recognition tool called HVite that allows recognition using language models and lattices. 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.

HVite uses a token passing algorithm like the one described in the previous chapter to perform Viterbi-based speech recognition. HVite takes as input a network describing the allowable word sequences, a dictionary defining how each word is pronounced and a set of HMMs. It operates by converting the word network to a phone network and then attaching the appropriate HMM definition to each phone instance. Recognition can then be performed on either a list of stored speech files or on direct audio input.

The word networks needed to drive HVite are stored using the HTK standard lattice format. This is a text-based format and hence word networks can be created directly using a text-editor. However, this is rather tedious and hence HTK provides two tools to assist in creating word networks. HBuild and HParse . Whichever method is chosen to generate a word network, it is

useful to be able to see examples of the language that it defines. The tool HSGen is provided to do this. It takes as input a network and then randomly traverses the network outputting word strings. HSGen can also compute the empirical perplexity of the task.

HDecode is a decoder suited for large vocabulary speech recognition and lattice generation that is available as an extension to HTK, distributed under a slightly more restrictive license. Similar to HVite, HDecode transcribes speech files using a HMM model set and a dictionary. The best transcription hypothesis will be generated in the Master Label File (MLF) format.

HDecode expects lattices where there are no duplicates of word paths. However by default lattices those are generated by HDecode contains duplicates due to multiple pronunciations and optional inter-word silence. To modify the lattices to be suitable for lattice rescoring HLRescore should be used to merge (using the -m option) multiple paths.

HDecode has the following limitations:

- ✓ Only works for cross-word tri-phones
- ✓ sil and sp models are reserved as silence models and are, by default, automatically added to the end of all words in the pronunciation dictionary.
- ✓ Lattices generated with HDecode must be merged to remove duplicate word paths prior to being used for lattice rescoring with HDecode and HVite.

2.8.4 Analysis Tools

A tool called HResults compares recognition results with original transcriptions. 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: error one an ***insertion error*** which occurs when the ASR system generates a word that does not

correspond to any word in the reference transcript, Error two a ***deletion error*** which occurs when the reference transcript contains a word that has no corresponding word in the ASR hypothesis and Error three a ***substitution error*** which 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*). Where *N* is the total number of labels in the reference transcriptions; the percentage of correct word recognized is then:

$$\text{Percent Correct} = \frac{N - D - S}{N} * 100\% \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 2.27$$

As we can see from the above expression it ignores the insertion errors. But the percentage accuracy defined by the below given expression includes insertion errors, and it is a more representative figure of recognizer performance.

$$\text{Percent Accurate} = \frac{N - D - S - I}{N} * 100\% \quad \ldots\ldots\ldots\ldots\ldots\ldots 2.28$$

The tool HResults outputs both of the above measures during result analysis.

The HTK tools, whenever used where they are designed for (data preparation, Training, Testing and Analysis) they take some inputs like files and scripts then returns the output after some processing.

2.9 Spontaneous speech ASR previous works

Spontaneous speech recognizer for Japanese developed by Furui S, et.al [26] using two different corpuses, One "Corpus of Spontaneous Japanese (**CSJ)"** speech: A part of the corpus completed by the end of December 2000, consisting of 610 presentations (approximately 1.5M words of 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 using 338 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 language 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 words [26].

Acoustic model developed using CSJ and acoustic model developed using read speech evaluated using test data set. 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 better matching of recording conditions with the test set.

Furui, et.al [27] 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.

Bantegize [28] 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 the experimental result, Bantergize [28] 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.

CHAPTER THREE

AMHARIC LANGUAGE

3.1 Background

Amharic is the working language of Ethiopia federal government. It is a Semitic language family that has the largest number of speakers after Arabic. It is spoken, as per the 1998 census, by 17.4 million people as a mother tongue and 5.1 million people as a second language. Solomon [6] Claims as Amharic has five dialectical variations spoken in different Amharic regions: Addis Ababa, Gojjam, Gonder, Wollo, and Menz.

Amharic is a member of the Ethio-Semitic languages, which belong to the Semitic branch of the Afro-Asiatic super family, and uses the Ethiopic script commonly, known as Ge'ez, which is an ancient language currently, used for the liturgy of the Ethiopian Orthodox Church.

Amharic uses thirty-three consonant classes form 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 [29]. Each consonant has seven forms created by fusion of a consonant and a vowel-fused form of vocalic marker. According to [30] 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. 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 [29]. 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 sub units of the Amharic language. The smallest unit of one language is its phone. Amharic has its own phonemes. Before describing Amharic phonetics, phonology, syntax and writing system it is appropriate to view the basic terms.

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 [6].

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.

3.2 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 [31]. 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 Juang and Furui these 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 [6].

Sounds can also be classified as vowels and consonants. Vowel and consonant sounds are produced in fundamentally different ways. According to [30] while consonants are articulated with a substantial degree of obstruction in the oral cavity, vowels are produced with a relatively free airflow.

3.2.1 Articulation of Amharic Consonants

According to Voicing, Manner and Place of articulation the following summary describes Amharic consonants.

Manner of Articulation	**Voicing**	**Place of Articulation**				
		Labials	**Dentals**	**Palatals**	**Velars**	**Glottal**
Stops	Voiceless	ፕ [p]	ት[t]	ች[tʃ]	ክ[k]	ኧ[ʔ]
	voiced	ብ[b]	ድ[d]	ጅ[dʒ]	ግ[g]	
	Glottalized	ጵ[p’]	ጥ[t’]	ጭ[tʃ’]	ቅ[q]	
	Rounded					
Fricatives	Voiceless	ፍ[f]	ስ[s]	ሽ[ʃ]		ህ[h]
	voiced		ዝ[z]	ዥ[ʒ]		
	Glottalized		ጽ[s’]			
	Rounded					
Nasals	voiced	ም[m]	ን[n]	ኝ[ŋ]		
Liquids	voiced		ል[l] ር[r]			
Semi-vowels	voiced	ው[w]			ይ[j]	

Table 3.1 Categories of Amharic Consonants adapted from [6]

Furthermore, as described on the work of [38] Amharic has four labialized consonants which are classified as labio-Velar on the work of [26] that are pronounced with a slight rounding of the lips. These are ጕ [g^w], ኵ [k^w], ቈ [q^w], and ኍ [h^w].

3.2.2 Articulation of Amharic Vowels

Vowels are open sounds, made largely by shaping the vocal tract rather than by interfering with the flow of air stream [29].

Vowels are most usefully described in terms of the position of the tongue as they are articulated. A vowel articulated with the body of the tongue relatively forward is classified as a *front vowel*; one made with the body of the tongue relatively high is a *high vowel*. Vowels produced with the body of tongue neither high nor low are called mid vowels. Vowels produced with the tongue body front are called front vowels while those made with the tongue body back are called *back vowels.* Those vowels made with the tongue body neither front nor back are called central vowels [29].

Vowels accompanied by lip rounding as in (u and o) are called rounded vowels while the other vowels are called unrounded vowels. Amharic has 7 (seven) vowels their description is given in the table below.

	Front/Unrounded	**Central/Unrounded**	**Back/Rounded**
High	ኢ[i]	እ[I]	ኡ[u]
Mid	ኤ[e]	ኧ[E]	ኦ[o]
Low		ኣ[a]	

Table 3.2 Categories of Amharic vowels adapted from [11]

3.3 Amharic Writing System

The Amharic script is an abugida, and the graphs of the Amharic writing system are called *fidel(ፊደል).* Each character represents a consonant + vowel sequence, but the basic shape of each character is determined by the consonant, which is modified for the vowel. Some consonant phonemes are written by more than one series of characters: /ሀ/, /ሰ/, /ጸ /, and /ሠ/ (the last one has *four* distinct letter forms). This is because these *fidel* originally represented distinct

sounds, but phonological changes merged them. The citation form for each series is the consonant+ ኧ form, i.e. the first column of the *fidel*.

There are disagreements among scholars whether the Amharic writing system is syllabic or alphabetic [6]. According to Clark et al., writing systems in which one symbol represents one syllable are called syllabic while writing systems in which one symbol represents one sound segment are called alphabetic. According to Bender, Cowley, Mullen argue that the Amharic writing system is syllabic while others [32], [29] say that it is not syllabic.

From the point of view of speech recognition Amharic writing system has some limitations. According to [6] some of the limitations of the Amharic writing system are:

- ✓ There are several duplicate characters that are normally used interchangeably
  For example: [ሀ, ሃ, ሐ, ኻ, ሓ, ኃ],[ሰ, ሠ],[ጸ, ፀ],[አ, ዓ, ዐ, ኣ]
- ✓ It lacks a mechanism to mark gemination of consonants.
- ✓ There exists ambiguity between sixth-order symbols with and without vowel in different contexts.

Grammatical Arrangement

We will not make a detailed explanation of the Amharic language's grammatical arrangements as it is beyond the scope of this study. We will cover the top level grammatical and morphological structure of the language in this section. The Amharic language has been declared to have word categories as **ስም**(noun), **ግስ** (verb), **ቅፅል** (adjective), **ተውሳከ ግስ** (Adverb), **መስተዋድድ**(preposition), and **ተውላጠ ስም (**pronoun) [29].

Noun: a word will be categorized as a noun, if it can be pluralized by adding the suffix ኦች/ዎች and used as nominating something like person, animal, and so on.

Verb: any word which can be placed at the end of a sentence and which can accept suffixes as/ህ/,/ሁ/,/ሽ/, etc. which is used to indicate masculine, feminine, and plurality is classified as a verb.

Adjective: any word that qualify a noun or an adverb, which actually comes before a noun**(e.g. ንበዝተማሪ)** and after an adverb**(በጣምንበዝ).** Other specific property of adjectives is, when pluralized, it will repeat the previous letter of the last letter for the word **(e.g. ረዥም--› ረዣዥም).**

Adverb: it will be used to qualify the verb by adding extra idea on the sentence. The Amharic adverbs are limited in number and include **ትናንት, ገና, ዛሬ, ቶሎ, ምንኛ, ክፉኛ,እንደገና, ጅልኛ,** and **ግምኛ.**

Preposition: preposition is a word which can be placed before a noun and perform adverbial operations related to place, time, cause and so on; which can't accept any suffix or prefix; and which is never used to create a new word. It includes **ከ፣ለ፣ወደ፣ስለ፣እንደ…**

Pronoun: this category further can be divided as deictic specifier, which includes **ይህ, ያ, እሱ, እሰዋ, እኔ, አንተ, አንች…;** quantitative specifier, which includes **አንድ, አንዳንድ, ብዙ, ጥቂት, በጣም…;** and possession specifier such as **የእኔ, የአንተ, የእሱ…**

The Amharic basic sentence is constructed from noun phrase and verb phrase **(ስማዊሀረግ+ ግሳዊሀረግ).**

Sentence= noun_phrase + Verb_Phrase.

For example, the sentence: **"ሁለት ትልልቅ ልጆች ትናንት በመኪና ወደ ጎጃም ሄዱ።"** have noun phrase **""ሁለት ትልልቅ ልጆች"** and verb phrase **"ትናንት በመኪና ወደ ጎጃም ሄዱ".**

The statement **(አረፍተ-ነገር)** will have the noun phrase and verb phrase combinations. The noun phrase and the verb phrase further will be divided to different particles such as other sub noun phrase and verb phrase, noun, adjectives, specifier and so on.

Amharic punctuation marks and numerlas

The Amharic documents collected should be pre-processed before the succeeding ASR components perform further operations. The punctuation marks are here discussed as it will help in pre-processing documents.

Similarly, numerals have greater impact on ASR systems. Since numbers are stored in different formats, there should be some kind of standardization that will help for training acoustic model.

If numbers can't be normalized to one standard, a document that has different number representation will remain irrelevant for the training of acousitc model.

In Amharic, there are different punctuation marks used for different purposes. In the old scripture, a colon (**ሁለት ነጥብ**፡) has been used to separate two words. These days the two dots are replaced with whitespace. An end of a statement is marked with four dots **(**አራትነጥብ።) while **ነጠላሰረዝ**(፣ or ፥) is used to separate lists or ideas just like the comma in English.

In Amharic, numbers can be represented using Arabic symbols. It has also its own number representations, Ethiopic number representations. Similarly numbers can be represented in a word alphanumerically. Table 3.1 shows the Arabic, Amharic and alphanumeric representation of numbers from[32].

Arabic	Ethiopic	Alphanumeric	Arabic	Ethiopic	Alphanumeric
1	፩	**አንድ**	20	፳	**ሃያ**
2	፪	**ሁለት**	30	፴	**ሰላሳ**
3	፫	**ሦስት**	40	፵	**አርባ**
4	፬	**አራት**	50	፶	**አምሳ/ሀምሳ**
5	፭	**አምስት**	60	፷	**ስልሳ/ስድሳ**
6	፮	**ስድስት**	70	፸	**ሰባ**
7	፯	**ሰባት**	80	፹	**ሰማኒያ**
8	፰	**ስምንት**	90	፺	**ዘጠና**
9	፱	**ዘጠኝ**	100	፻	**መቶ**
10	፲	**አስር**	1000	፲፻	**ሺ/ሽህ**

Table 3.3.: Number Representations in Amharic adapted from [32]

Since numbers in Amharic during speech is pronunced not like the exact representaion it must be pre-processed as it is shown on table 3.4.

Fraction	Amharci representation	Ordinals	Representation
½	ግማሽ	1st	አንደኛ/ቀዳማዊ
1/3	ሲሶ	2nd	ሁለተኛ/ዳግማዊ
¼	ሩብ/እርቦ	3rd	ሶስተኛ/ሳልስ
2/3	ሁለትሲሶ / ሁለትሶስተኛ	4th	አራተኛ/ራብዕ
¾	ሶስትአራተኛ		
1/10	አስራት		
2X	እጥፍ	9th	ዘጠንኛ/ዘጠነኛ
2.X	ሁለትነጥብ X	10th	አስረኛ

Table 3.4 Amharic fraction and Ordinal representation adapted from [32]

On the above table we can easily observe that in the collected text if there exist fraction and ordinal representations are found it is normalised into the equivalent Amharic representation.

Gemination

As in most other Ethiopian Semitic languages, gemination is contrastive in Amharic. That is, consonant length can distinguish words from one another; for example,አለ (*alä)* 'he said', አለ (*allä)* 'there is';ይመታል(*yəmätall)* 'he hits', ይመታል (*yəmmättall)* 'he is hit'. Gemination is not indicated in Amharic orthography, but Amharic readers typically do not find this to be a problem. This property of the writing system is analogous to the vowelsof Arabic and Hebrew or the tones of many Bantu languages, which are not normally indicated in writing.

The noted Ethiopian novelist HaddisAlemayehu, who was an advocate of Amharic orthography reform, indicated gemination in his novel ፍቅርእስከመቃብር (*FəqərƏskäMäqabər)* by placing a dot above the characters whose consonants were geminated, but this practice is rare. Forexmple: " እኔና' : እግዚአብሄር : የምናውቀው : እርስዎ : የማያውቁት : አለ' : አባቴ ? (ፍቅርእስከመቃብር page ፲፪)From the above sentence there are two words that are geminated the word እኔና(Inenna) and አለ(*allä*) from the two words the consonant 'ና' in the word እኔና is geminated and it makes the word to give another meaning እኔና'(Inenna) 'me and' when it is geminated but if it was not geminated it will give another meaning እኔና (Inena) 'me come' and the word አለ'(*allä*) 'there is' is geminated but if it was not geminated it will give another meaning አለ (*alä)* 'he said' and it will give another meaning in the sentence.

When we Geminate the words1 : “እኔና’: እግዚአብሄር : የምናውቀው : እርስዎ : የማያውቁት : አለ’ : አባቴ ?”

Tanslated equivalent result: “father, is there anything that me and God knows that you don’t know?”

When we don’t geminate the words: “እኔና : እግዚአብሄር : የምናውቀው : እርስዎ : የማያውቁት : አለ : አባቴ?”

Translated equivalen result :“my father says me and God knows that you don’t know”

The ortographic representation of Amharic sentence is the same whether we geminate it or not that is why Haddis Alemayehu use dot on the top of the gemminated consonant symbolto mark the geminated consonant.As we can see from the above examples since gemination results in degradation of speech recognition results it is better to use the rule of gemination by inserting special symbols to identify the geminated words that are uttered.

CHAPTER 4

AMHARIC SPONTANEOUS SPEECH ASR PROTOTYPE

With this study, in order to develop the Amharic spontaneous speech recognition system different experiments has been carried out. In this chapter major tasks and components of ASR used in statistical approach including data preparation, training, testing of recognizer and analysis of recognizer performance are described. With this chapter the analysis and discussion of the results found are also included. The HTK tools used and their output are given with figure 4.1 which depicts the developed system architecture.

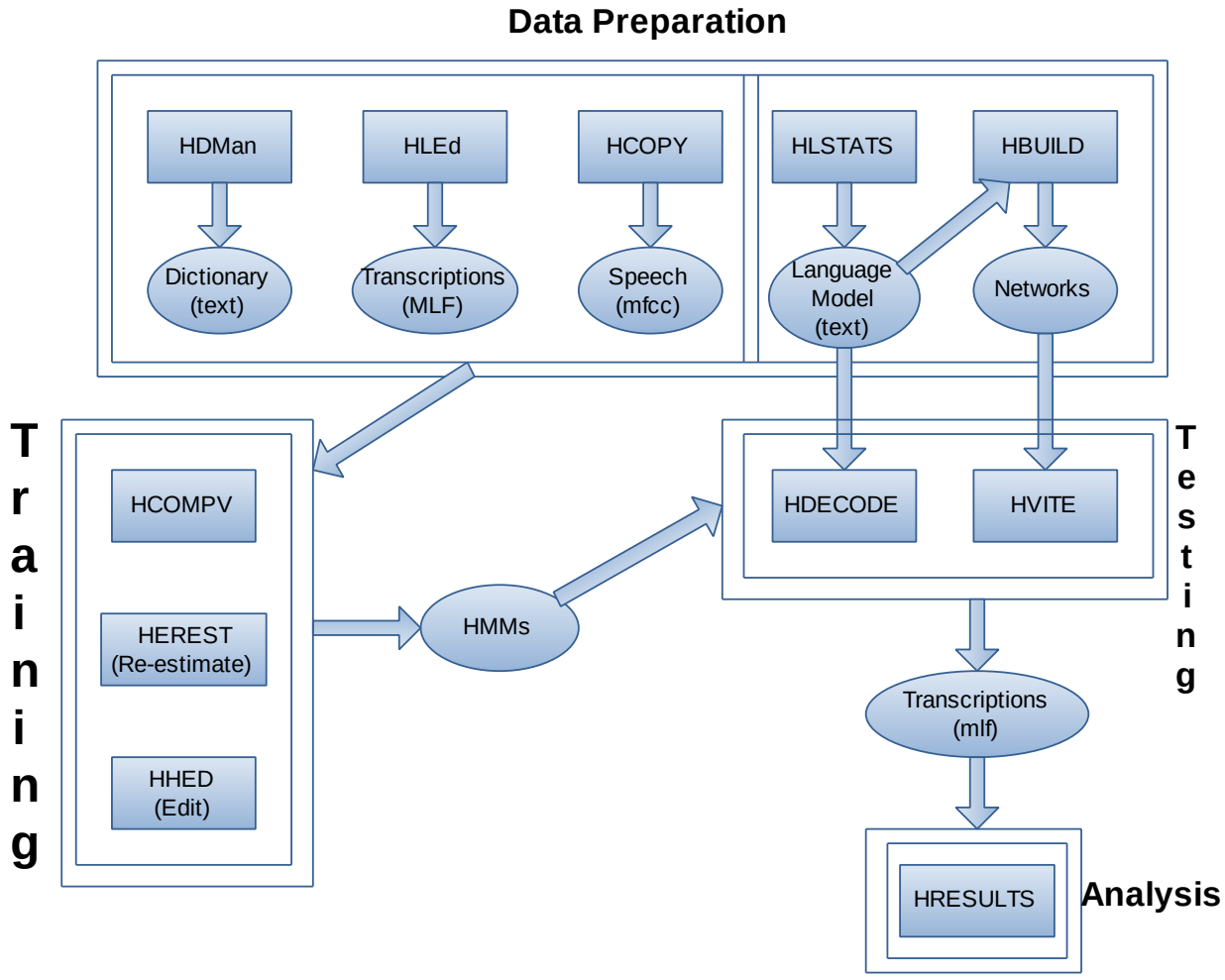

Figure 4.1: Architecture of the system

Chapter 4: Amharic spontaneous speech ASR prototype

4.1 Data preparation

Data preparation is the first task that should be addressed in speech recognition development process. For this study, the speech data with 44100 Hz sampling rate obtained from web that are recorded in local medias. The speech data used for both training and testing are conversational speeches which are made between two or more speakers in the form of conversation. Like any other spontaneous speech our speech data have different non-speech events that have effects on the performance of recognizer. Depending on these non-speeches frequency we have prepared two types of data for both training and testing. One by modeling all non-speeches found while we transcribe the speech, the second data set is prepared by modeling more frequent non-speech events.

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, in average 56 sentences 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 test data is constructed from, both speakers who are involved in the training and not involved in the training. **Test** data have 14 total numbers of speakers and it involves utterances of 9 male speakers and 5 female speakers. The numbers of words in this test data are around 820 unique words which are from 104 utterances (sentences) and it is around 10 minutes long.

Both training and testing data are prepared by collecting audio files from local Medias. But the transcriptions (the orthography) of these audio files are manually written by the researchers. Since the speech recognition tool which is implemented for this study HTK does not support Amharic text as they are, the transliteration is done. This transliteration is done by converting the Amharic alphabets into their equivalent IPA representations. With a little modification adapted from [6].

በአድዋ bä?adɨwa ሰዓትsä?atɨ

አዲሳባ?adisaba ጨመርኩኝCämärɨkuɲɨ

Chapter 4: Amharic spontaneous speech ASR prototype

Spontaneous speech, in addition to normal speeches it contains some non-speech events. These non-speech events can be filled pauses, lip smack, breathing, long pause, repetitions, hesitations and many others. In the process of speech recognizer development since they have effects, not only normal speeches but also these non-speech events should be modeled. So like normal speeches we have to transcribe them.

Generally there are different types of non-speech events but in this paper we have transcribed and modeled only some of them which are observed in our speech data. Filled pause (FP), hesitation (HES), breathe (BR), Other Speaker (OTH), Throat clear (THC), Laugh (LGH), Lip smack (LIP), repetition and Interruption (INT) of other speaker are non-speech events which are transcribed and modeled in this study. Non speech events like background noise and breathe at the end and start of the sentence are not modeled since they can be tolerated by the tool used.

Depending on their sources, these non-speech events that occur in speeches can be from one of below given three major sources. (1) Speaker Generated (2) Other Speaker Distortions (3) Background Noise.

1. **Speaker-generated non-speech events**

Speaker generated non-speech events are Hesitation (HES), Lip smack (LIP), Repetition, Filled Pause (FP), Throat Clear (THC), Breathe (BR) and Laugh (LGH). Since they are produced by the speaker the same way as speech, Speaker-generated non-speech events always occur between words and sometimes in a rare case with words like in the case of speaking while laughing. They can be therefore annotated as another word, using a symbols assigned for each of them (e.g. if there is filled pause between two words, ‘word1 FP word2’).

FP መጀመሪያ FP ሚኒሊክ ትምህርት ቤት የሰራሁት FP መነባነብ ነው

[FP mäyämärija FP minilikɨ tɨmɨhɨrtɨ betɨ yäsärahutɨ FP mänäbanäbɨ näwɨ]

Repetitions of the same words are also common challenges in the natural informal speech which is not read. The repeated words are sometimes contextually important but sometimes the speakers’ simply repeat the word (words) which do not have any change on the meaning. Sometimes they may repeat partial part of the word in order to correct it, in this case it is

transcribed as a hesitation. But at the time of complete repetition of the word which does not bring any change on the meaning of the sentence, it is identified by '~' symbol. But the pronunciation dictionary of the word is assigned as a normal word.

እሱ~እሱእንዴትመጣናአየሽደራሲው[?ɨsu~?ɨsu ?ɨnɨdetɨmäTana ?ajäSɨdärasiwɨ]

ከዛን ጊዜ *ከዛን ጊዜ* ጀምሮ በቃ ያው አቆምኩት[käzanɨ gize ~käzanɨ ~gize dʒämɨro bäqa jawɨ ?aqomɨkutɨ]

Due to the naturalness of spontaneous speeches the irregular lengths of words are exist. This irregular length is modeled by duplicating four times the vowel which is lengthening irregularly. Although the extra vowels added in order to show its irregular lengthiness, the pronunciation dictionary for the word is considered as a normal word.

<u>የ</u>ሁለት አመት ልጅ ነው የደወለቺው [yäääähulätɨ ?amätɨ lɨdʒɨ näwɨ yädäwäläciwɨ].

The word 'yäääähulätɨ' is stretched abnormally.

2. Other speaker in background

As the speech is from interview in different environments, the audience present in the room can influence the speech and distortion from other speaker can be present. Even if no audiences in the room the interviewee and the interviewer can speak simultaneously. We can resolve two different situations here, either the distortion appear within speech pause (OTH) or more speakers are talking simultaneously like in case of cocktail. There is no situation when many speakers speaking simultaneously. But we have resolved the distortion appear within speech pause particularly between words.

ቀበድ አለው ወይ OTH እያልኩ አሾፍበታለው[qäbɨdɨ ?aläwɨwäjɨ OTH ?ɨyalɨku ?aSofɨbätaläwɨ]

3. Background Noise events

Environmental distortion can overlap particular words, so we need to use special rules for better description of noise disturbance within the speech. If the noise appears only in the pause between words, it is marked similarly to speaker-generated events. When the following speech is affected,

starting (INT-) and ending (-INT) mark is used like, e.g. “word1 INT− word2 –INT word3 INT-word4-INT”.

ከህዝብ ጋር በደንብ ተገናኘሁ ማለት ነው[INT-kähizibi-INT garɨ bädänɨbɨ tägänaɲähu malätɨ näwɨ]

After we have transcribed speeches which are collected both for training and testing purpose, we have proceeded to another data preparation tasks using HTK and another tools accordingly.

4.1.1 Pronunciation Dictionary

To create a pronunciation dictionary in HTK we need to have a word list (**wordlist**) which is a sorted list of the unique words that appear in the utterances (sentences) file. For this study we derived a list of unique words from sentences using the Perl script **prompts2wlist** which is available with HTK sample directory**.**

Pronunciation dictionary developed (*speech_dict.lex*) using python code we have prepared for this purpose. The transliteration was done by converting all the Amharic phones as a consonant and vowel combination for all phones. But the Amharic alphabets sixth order phones can be combined with vowels or the consonant can be written alone depending on the contextual word.

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.

HDMan -m -w wordlist -n monophones1 -l dlog dict speech_dict.lex nonspeech_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 pronunciation dictionary.

The general format of each dictionary entry created by *HDMan* command 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.

For irregular length words, although the extra vowelis added in order to show its irregular lengthiness, the pronunciation dictionary for the word is considered as a normal word.

የሁለት አመት ልጅ ነው የደወለቺው [yEEEEhulEtI HamEtI lIDI nEwI yEdEwElEciwI][1].
Although the word, 'yEEEEhulEtI' is lengthened its dictionary is normalized.

yEEEEhulEtI [yEhulEtI] y E h u l E t I sp

Except some special cases are considered with some of them, non-speech events are treated as a word and their pronunciation dictionary is assigned. As it is explained above, at the time of complete repetition of the word which does not bring any change on the meaning of the sentence, it is identified by '~' symbol. But the pronunciation dictionary of the word is assigned by concatenating the normal phones like a normal word.

እሱ~እሱእንዴትመጣናአየሽደራሲዉ [HIsu~HIsu HInIdetI mETana HayESI dErasiwI]

~HIsu [HIsu] H I s u sp

HIsu [HIsu] H I s u sp

The vocabulary (words) used for Training in this experiment, excluding sp, sil and phones assigned for non-speech, it consists of 36 total Amharic phones.

4.1.2 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, we have used a

[1] IPA Amharic phones changed during transliteration: [IPA=changed in to]
ʧ=C, ʧ'= c, ʧ`=c , t'=T, ʃ=S, s'=x , p'=P, ɲ=N, k'=q , ʤ=D, j=y, ʒ=Z

flat-start scheme. To do this, two sets of phone transcriptions needed. The set used initially do not have short-pause (*sp*) models between words. Then once reasonable phone models have been generated, an *SP* model will be inserted between words to take care of any pauses introduced by the speaker.

Since HTK toolkit cannot process prompts (sentences) 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. Since the second one is the easiest approach we have preferred it for our experiment. In order to generate the *mlf* file from our prompts, we have used the Perl script **prompts2mlf**which is provided with HTK samples.

```
$perl ../HTK_scripts/prompts2mlf  words.mlf   train_prompts
```

This script generates a word level transcription, ***words.mlf*** file.

Phone Level Transcriptions

After word level transcription we executed the **HLEd** command which is provided with HTK tool to expand the **Word Level Transcriptions** to **Phone Level Transcriptions**. This command replaces each word by its equivalent phonemes 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 dictionary file we created earlier, and out putting the result in a file called **phones0.mlf** which do not have short pauses ("SP"s) after each word phone group.

Before executing this command first, we have created the ***mkphones0.led*** edit script with the following content:

```
EX
IS silsil
DE sp
```

Then executing the HLEd command creates the ***phones0.mlf*** file.

```
$HLEd -A -D -T 1 -l '*' -d dict –I  phones0.mlf  mkphones0.led words.mlf
```

Next, the second **phones1.mlf** file which includes short pause ("sp") after each word phone group created. In this case also before the execution of the command we have created the ***mkphones1.led*** edit script with the following contents:

```
EX

IS sil sil
```

Then we run the HLEd command and it creates ***phones1.mlf***filefor us.

```
$HLEd -A -D -T 1 -l '*' -d dict –I  phones1.mlf mkphones1.led words.mlf
```

4.1.3 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 another format. HTK 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.

We use the HCopy tool to convert our wav files to MFCC format. We have 2 options. We could execute the HCopy command by hand for each of our audio (wav) files, or we can create a file containing a list of each source audio file and the name of the MFCC file it will be converted to,

and use that file as a parameter to the HCopy command. We will use the second approach in this experiment. To do this we create the ***'codetrain.scp'*** script file, see at Appendix B.

The HCopy command performs the conversion from wav format to MFCC. To do this, a configuration file which specifies all the needed conversion parameters is required. For this purpose we have prepared the configuration file (hcopy_config) with different parameters. Different tools configuration file are given under Appendix C.

Some of these settings specify that, the source format is WAV file, the target parameters are to be MFCC using C0 as the energy component, the frame period is 10msec (HTK uses units of 100ns), the output should be saved in compressed format, and a *crc* checksum should be added. The FFT should use a Hamming window and the signal should have first order pre-emphasis applied using a coefficient of 0.97. The filter bank should have 26 channels and 12 MFCC coefficients should be output. The variable ENORMALISE is by default true and performs energy normalization on recorded audio files. It cannot be used with live audio and since the target system can be used for live audio, this variable is set to false.

Then executing HCopy command as follows creates a series of *mfc* files corresponding to the audio files as it is listed in **codetrain.scp**.

```
$HCopy -A -D -T 1 –C  hcopy_config  -S  codetrain.scp
```

4.2 Training the model

4.2.1 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 phone-based system, and we have done training with topology 5-state (3 emitting states) left to right with no skips and with skips, 7-state (5 emitting states) left-right with no skips and with skip. 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 D.

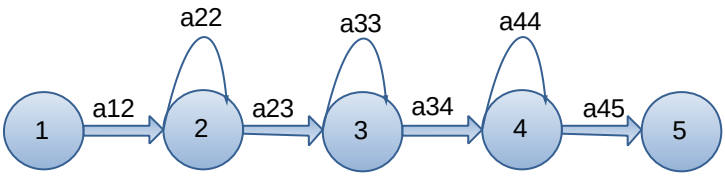

Figure 4.2 HMM model with 3 emitting state

HMM model with 3 emitting state and with skip is given in figure 4.3.

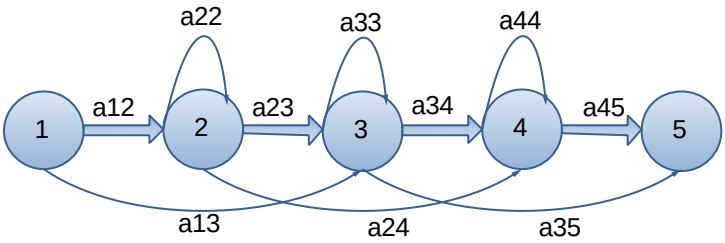

Figure 4.3 HMM model with 3 emitting state and with skip

The HTK tool HCompV will scan a set of data files, compute the global mean and variance and set all of the Gaussians in a given HMM to have the same mean and variance.

HCompV -A -D -T 1 -C config -f 0.01 -m -S train.scp -M hmm0 proto

Hence, assuming that a list of all the training files is stored in ***train.scp,*** the above command creates the new version of our **proto** file in which the zero means and unit variances are replaced by the global speech means and variances.

Using new prototype model generated by HCompV, a Master Macro File (MMF) called *hmmdefs* containing a copy for each of the required mono-phone HMMs is constructed by manually copying the prototype and relabeling it for each required mono-phone (including "sil"). The

format of an MMF is similar to that of an MLF and it serves a similar purpose in that it avoids having a large number of individual HMM definition files.

The file *macros* contain a global options macro and the variance floor macro vFloors generated earlier by HCompV. The global options macro simply defines the HMM parameter kind and the vector size.

4.2.2 Re-estimating mono-phones

The mono-phones 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

The above command loads all the models both *hmmdefs* and *macros* which are listed in the model list. Mono-phones 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.

The -t option sets the pruning thresholds to be used during training. Pruning limits the range of state alignments that the forward-backward algorithm includes in its summation and it can reduce the amount of computation required by an order of magnitude. For most training files, a very tight pruning threshold can be set, however, some training files will provide poorer acoustic matching and in consequence a wider pruning beam is needed. HERest deals with this by having an auto incrementing pruning threshold.

In the above command, pruning is normally 250.0. If re-estimation fails on any particular file, the threshold is increased by 150.0 and the file is reprocessed. This is repeated until either the file is successfully processed or the pruning limit of 1000.0 is exceeded.

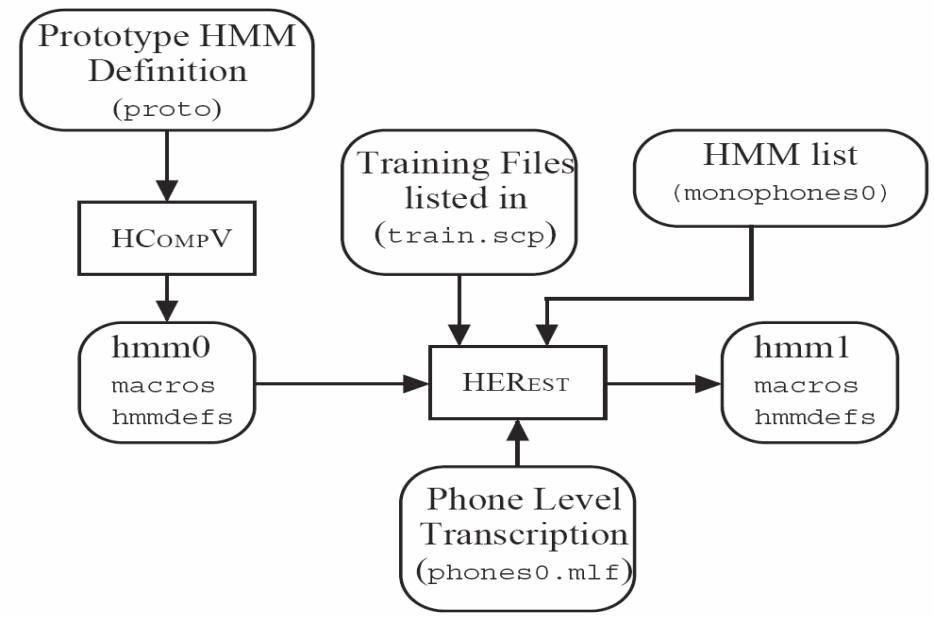

Figure 4.4 Creating flat-start mono-phones adapted from [25]

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 Fig. 4.5.

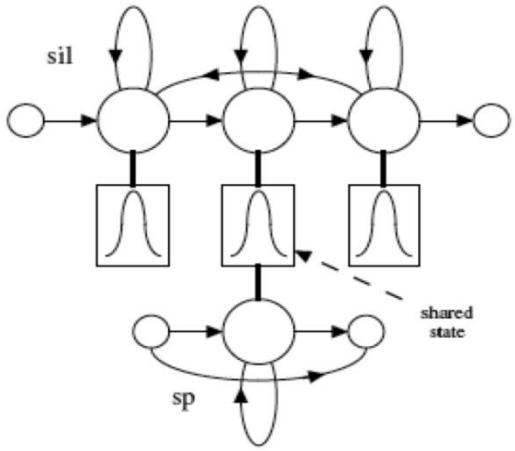

Figure 4.5 Silence Models Adapted from [25]

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:

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

Where sil.hed contains the following commands:

```
AT 2 4 0.2 {sil.transP}
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.

Realigning the Training Data

The key difference between this operation (realigning) and the original word to-phone mapping performed by HLEd in data preparation is that, in this operation all pronunciations for each word are considered and outputs the pronunciation that best matches the acoustic data. Using the phone models created so far we have realigned the training data and creates new transcriptions with a single invocation of the HTK recognition tool HVite:

HVite -l '' -o SWT -b SENT-END -C config -a -H hmm7/macros -H hmm7/hmmdefs -ialigned.mlf -m -t 250.0 -y lab -I words.mlf -S train.scpdict monophones1*

This command uses the HMMs made previously to transform the input word level transcription *words.mlf* to the new phone level transcription *aligned.mlf* using the pronunciations stored in the dictionary. When aligning the data, it is sometimes clear that there are significant amounts of silence at the beginning and end of some utterances, so to spot this the time-stamp information need to be output during the alignment. That is why we have used the option *-o SW* in the above command. After the new phone alignments have been created, HERest applied to re-estimate the HMM set parameters.

4.2.3 Refinements and Optimization

A. Tied-State Tri-phones

As stated by [25] 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. Butbefore 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 to be 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.

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 be 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.mlf mktri.led aligned.mlf*

Using the above command we have created the tri-phone transcriptions in *wintri.mlf* file using mono-phone transcriptions *aligned.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
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 triphones (and biphones) 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

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"
```

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 '*R_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 Kinfe [11].

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 triphones. The effect of the AU command is to use the decision trees to synthesize all of the new previously unseen triphones 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.mlf xwrd.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 sil sil
RE sp sp
```

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 *itemList*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}*

C. Increasing number of state

In order to improve the performance of the recognizer, the emitting state of the model has been increased from three to five. To do this we have prepared the new prototype file which enables us for flat starting. The prototype used file is given at Appendix E. the HMM topology with 5

emitting state shown in figure 4.6. As we can see it from this figure the first and the last states are non-emitting states.

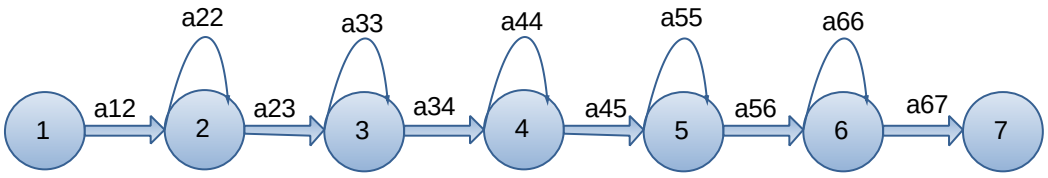

Figure 4.6 HMM model with 5 emitting states

D. Modeling Non-speech events

Size of data has significant effect on performance of recognizer. As the size of data increases the performance also increases. For speech recognition which their units of recognition are sub-words (mono phone, tri-phone, syllables and etc.) the frequency of these recognition units directly affects the recognizer. Phones which have more frequency in training have better probability to be recognized by the recognizer. In case of spontaneous speech in addition to sub-word units, the frequency of non-speech events also have effect on the recognizer. Since we have modeled them in the development of the recognizer, like phones their frequency has impact on recognizer. The List of non-speech events and their frequency which observed in our data are given in table 4.1.

Non-speech events	**Frequency**	**Remark**
Filled Pauses (FP)	1114	
Breathing (BR)	656	
Hesitation (HS)	240	
Lip Smack (LIP)	82	
Repetitions (~)	33	
Throat clear (THC)	3	
Laughing (LGH)	10	
Other Speaker (OTH)	138	Between words
Interruption (INT)	52	Occur concurrently

Table 4.1 Frequency of none speech events

Among none speeches observed during transcription we have not modeled Lip smack, throat clear, laughing and interruption in language model since they are less frequent. As we can observe from the above table Throat clear occurs only three times and laughing occurs only ten times; Related to others these non-speech events appeared less times than another none speeches found during transcription. Lip smack is also not modeled in this data set since in addition to its less in number, most of the times it comes after and before breathing and it were a challenge to differentiate from breathing; thus even it was difficult to transcribe them surely.

Another none speech which is discarded from this data set is Interruption, this occurs most of the times if there is any disturbance or noise at the back ground while the speaker speaks, this noise may be another speaker speaking simultaneously or noisy from another sources. while we model this none speech as we have explained it in data preparation section we have marked the words which are interrupted during utterance, here the words are clearly audible but we have marked it simply in order to indicate that there is interruption. By considering that the HTK tool may tolerate these events considering them as background noisy we have avoided them from this data set.

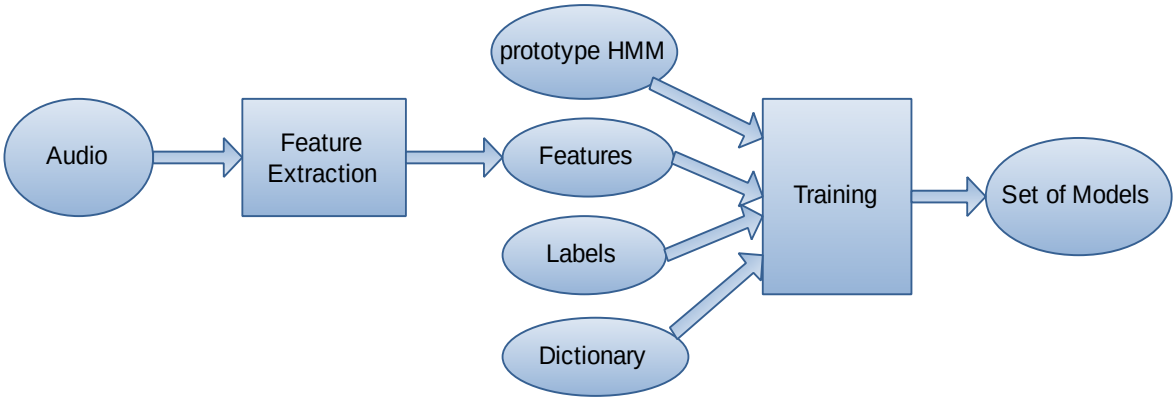

Figure 4.7 Summary of one time training process

Language Model

The Language model used for recognition was developed using 10,100 sentences both text corpus taken from Solomon [6] and training text corpus we have prepared for this study. The text data from Solomon [6] used to increase the text data size; since as the size of the text data

increases the probability of occurrence of words increases. This increment of word probability have improvement on speech recognizer accuracy .There is two HTK tools *HLStats* and *HBuild* which helps us for development of bigram language model and word network respectively.

HLStats is implemented to build a bigram language model;

HLStats –b bigram –o wordlist train.mlf

And then HBuild used to construct word-loop and word-pair grammars automatically and also it can incorporate a statistical bigram language model into a network.

HBuild –n bigram wdnet train.mlf

4.3 Recognizer testing and evaluation

4.3.1 Recognizing

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.

Assuming that *test.scp* holds a list of the coded test files, and then each test file recognized and its transcription output to an MLF file called *recout.mlf* by executing the following:

HVite -H hmm15/macros -H hmm15/hmmdefs -S test.scp–t 250.0 150.0 3000.0 -l '' –I recout.mlf -w wdnet -p 0.2 -s 15.0 dicttiedlist*

HDecode -H Models -S test.scp -t 200.0 300.0 -C config.hdecode–I xwrd recout.mlf -w bigram -p 0.02 -s 15.0 dict.hdecode treeg.list

The options -p and -s set the word insertion penalty and the 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 have a significant effect on recognition performance, so some modifications made and experimented with different parameters. We found relatively better recognition result using 0.2 and 15.0 for word insertion penalty and grammar scale factor respectively. The process of recognizing and knowledge sources used using HTK, is given in figure 4.7.

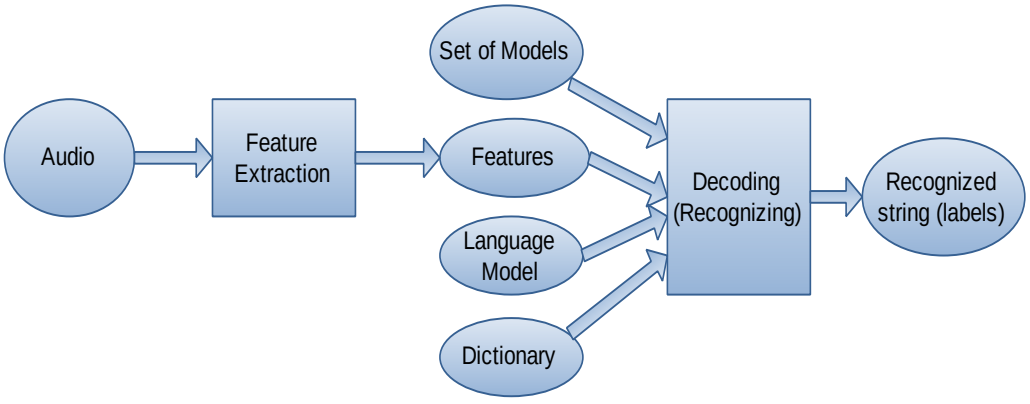

Figure 4.8 Summary of recognition process

4.3.2 Analysis

Once the test data has been processed by the recognizer, the next step is to analyze the results. The HTK tool HRESULTS is provided for this purpose. HRESULTS compares the transcriptions output by HVITE and HDECODE with the original reference transcriptions and then outputs various statistics. HRESULTS matches each of the recognized and reference label sequences by performing an optimal string match using dynamic programming.

Once it founds the optimal alignment, HRESULTS calculates the number of substitution errors (*S*), deletion errors (*D*) and insertion errors (*I*).then outputs both percentage of correctly recognized words and percentage of accurately recognized words. As with other HTK tools it can process individual label files and files stored in MLFs. Using our MLF file which contains word level transcriptions for test file, the actual performance is determined by running HResults as follows:

```
HResults –I testref.mlf tiedlist recout.mlf
```

4.4 Comparison of results 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.

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 but the recognizer performance starts decreasing at twelfth Gaussian mixture. Therefore all the results we have reported are the results found at eleventh mixture. The results are given below in tables and followed by discussion.

Topology	No of Gaussian Mixture	Cross-word Tied-state Tri-phone		Word internal tied state tri-phone	
		Percentage of words recognized	Word Accuracy percentage	Percentage of words recognized	Word Accuracy percentage
3 states without skip	11	39.17%	23.33%	45.87%	29.33%

Table 4.2 Results of cross-word and word internal tri-phones

In the process of this recognizer development we have developed the recognizer using context independent mono-phones and context dependent tri-phone. It is also stated by sub-word based Amharic ASR developers [11] [13], similarly our recognizer developed using context dependent tri-phone with tied states has better performance than the context independent mono-phone. During our attempt for enhancement from context independent phones (mono-phone) to tri-phone, we have developed two sets of tri-phones. The first set made up of word internal tri-phones and the second set contains cross-word tri-phones.

The recognizer developed using these two tri-phones was tested using the same bigram language model but HVite tool was used for word internal tri-phones while HDEcode tool was used for cross-word tri-phones. As we can observe from the above table the analysis result for word internal tri-phone is 29.33% and for cross word tri-phone is 23.33 %. The tri-phone which is word internal performs better. As we have mentioned under HTK recognizer tools section, HDEcode designed for large vocabulary, cross-word tri-phones and tri-gram language model. In contrary our data is small size and bigram language model.

Topology	No of Gaussian Mixture	Word internal tied state tri-phone	
		Percentage of words recognized	Word Accuracy percentage
3 states without skip	11	45.87%	29.33%
3 states with skip	11	24.32%	15.15%

Table 4.3: Results for 3 states with and without skip

As we can see from the table 4.3, the accuracy of a 3 emitting state HMM without skip is29.33% and with skip accuracy is 15.15%. This was due to the size of data, if some states skipped there are phones which were reduced and this causes decrement in data size. The performance of cross-word tri-phone was less than word internal tri-phones we have preceded our experiment by leaving cross-word tri-phone, using word internal tri-phones. The results we have found using word internal tri-phones have been given in tables and discussed in the following paragraphs.

	Word internal tied-state tri-phone (3 state without skip)		Word internal tied-state tri-phone (5 state without skip)	
Speaker code	Percentage of words recognized (%)	Word Accuracy percentage (%)	Percentage of words recognized (%)	Word Accuracy percentage (%)
tst001	55.64	12.78	53.38	19.55
tst002	58.33	47.22	54.63	49.07
tst003	70.21	42.55	53.19	36.17
tst004	50.81	33.06	57.26	47.58
tst005	50.91	35.45	50.00	37.27
tst006	61.42	44.09	59.84	44.88
*[2]tst007	35.48	17.74	40.32	19.35
tst008	54.04	37.87	51.10	41.18
tst009	57.14	42.86	54.55	37.66
tst010	33.72	25.58	47.95	32.88
tst011	48.21	28.57	56.25	44.64
*tst012	14.52	11.29	22.58	17.74
*tst013	15.38	7.69	30.77	15.38
*tst014	5.00	0.00	10.00	10.00
Average Performance	**45.87**	**29.33**	**43.92**	**34.54**

Table 4.4 Analysis of results when all non-speech events modeled

[2] Speakers not involved in training

The model with three emitting state HMM without skip, has word recognition accuracy of 29.33 % while the HMM model with five emitting state without skip and jump, has word recognition accuracy 34.54 %. Solomon [6] Reported that the recognizer developed using five emitting state performs better than the recognizer developed using three emitting state. In a similar way, as we can see from the given percentage of performance, our recognizer developed using HMM with 5 emitting state has better performance than HMM with 3 emitting state.

Speaker code	**Percentage of words recognized (%)**	**Word Accuracy percentage (%)**
tst001	52.63	17.29
tst002	57.01	49.53
tst003	59.57	31.91
tst004	52.03	44.72
tst005	50.91	37.27
tst006	62.99	50.39
*tst007	50.00	30.65
tst008	64.71	45.96
tst009	57.14	44.16
tst010	54.79	47.95
tst011	53.21	41.28
*tst012	30.65	17.74
*tst013	38.46	23.08
*tst014	10.00	5.00
Average Performance	**55.46**	**39.86**

Table 4.5 Results when most frequent non-speech events modeled

As we can see from the table 4.4 the word recognition accuracy result for the recognizer developed using data set which contains all observed non-speech events is 34.54% and as it is shown in table 4.5 the analysis result for recognizer which was developed using by including

only most frequent of non-speech events is 39.86%.The recognition accuracy which is found from the data set with only most frequent non-speech events modeled is better. As [33] stated that the spontaneous ASR performance degraded due to the occurrence of dis-fluencies with spontaneous speech, similarly our ASR recognition increases with the decrement of dis-fluencies.

The speech from speaker *tst002* was recognized 49.53% word accuracy while the speech from the speaker *tst003*was recognized 31.91% word accuracy. Even if there might be other factors which can depict the reason for the difference of these results, the largest sentence uttered by speaker *tst002* contains less number of words including non-speeches but the sentences uttered by speaker *ts003* consists of more number of words. This is one of the reasons why the recognizer more recognizes the speech from the first speaker than the later speaker. Furui et.al [34] Claims that the length of the uttered sentence is one of the factors which decreases accuracy of recognizer developed using spontaneous speech.

The result found when the most frequent non speech 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 high number of insertion error which is decreasing the accuracy of the recognizer. This insertion error occurs due to different reasons but to our knowledge in our experiment, it might be related to transcription of speech data in to text. During transcription since it is done manually, there might be words in speech data used for test but not exist in transcribed text data, the decoder recognizes the words in the speech data but at the time of analysis if we compare the recognizer output with our test data, since they do not present in the test data they are displayed as an insertion error.

Speaker code	For speakers involved in training	
	Percentage of words recognized (%)	**Word Accuracy percentage (%)**
tst001	52.63	17.29
tst002	57.01	49.53
tst003	59.57	31.91
tst004	52.03	44.72
tst005	50.91	37.27
tst006	62.99	50.39
tst008	64.71	45.96
tst009	57.14	44.16
tst010	54.79	47.95
tst011	53.21	41.28
Average Performance	**57.47**	**41.60**

Table 4.6 Recognition result for speakers involved in training

For speakers not involved in training		
Speaker code	**Percentage of words recognized (%)**	**Word Accuracy percentage (%)**
*tst007	50.00	30.65
*tst012	30.65	17.74
*tst013	38.46	23.08
*tst014	10.00	5.00
Average Performance	**39.79**	**23.25**

Table 4.7 Recognition result for speakers not involved in training

As we can see from the table 4.6 and 4.7, by using the test data from 10 speakers which are involved in training, the average performance of the recognizer found was 41.60% word

accuracy. The average performance of the recognizer found using the test data from combination of speakers those involved in training and not involved in training was 39.86% word accuracy. The average performance of the recognizer found for the test data from the speakers those do not involved in training was 23.25% word accuracy. The recognizer better performs for the test data from speakers involved in training. This difference comes from the fact that, the recognizer better identifies the feature of the speeches which are from the speakers involved in training.

4.5 Challenges

There are a lot of challenges we face regarding resources and in different stages across the study. The data preparation stage was the most challenging task for our study. As this is the pioneer study in the area of Amharic spontaneous speech recognition, there was no prepared spontaneous speech corpus for this purpose that we can use for training and testing, so we developed it from the scratch. In this process of data preparation since we did it manually we face challenges in the course of audio data collection, transcribing these audio data into text (orthography).

The speech data used for training and testing was conversational in addition to speech there are non-speech events which have impacts on speech recognizer performance. In order to optimize the performance of the recognizer handling these non-speeches and dis-fluencies was one of the difficulties which needs further work.

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 Conclusion

With this study, the possibility of the speaker independent spontaneous speech recognizer for Amharic language has been explored. To do this, we have reviewed different related literatures thoroughly on spontaneous speech automatic speech recognizer development, about ASR for Amharic language and many more related issues. For this study, among the approaches of ASR systems the Statistical (stochastic) approach was used and the HMM has been implemented for modeling, HTK tool and different tools which are compatible with the approach and modeling technique we have preferred were implemented accordingly.

The recognizer developed, Spontaneous speech recognizer for Amharic, has been done using, small speech data size from 36 speakers and it was around 3 hours and 10 minutes for training. The test data was prepared with 104 utterances consisting of 820 unique words from 14 speakers, which 10 of them involved in training and 4 of them did not involve in training. The language model used was bigram language model developed using HLStats and word networks built from this bigram using HBuild tool.

Non-speech events (dis-fluencies) which are the main distinguishers of spontaneous speech from read speech occurs in both training and test data. Due to their direct influence on the performance of recognizer, rather than treating them as a silence we have modeled them according to their occurrences as we have explained it under data preparation section.

The training has been done first by modeling context independent mono-phones 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 and re-estimated tri-phone 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 11^{th} Gaussian mixture.

Chapter 5: Conclusion and Recommendation

During the course of this study the recognizers developed using different acoustic models has been tested using test data we have prepared for this purpose. Tuning the parameters of the decoder at the time of recognition brought different recognition accuracy. Some of these parameters are word insertion penalty (p), grammar scale factor (s) and pruning level.

Using bigram language model, 0.2 insertion penalty, 15.0 grammar scale factor and 3000.0 pruning level and other conditions kept similar while running the decoder, our recognizer best performs with data set that contains only the model of most frequent non-speech events. Regarding the HMM state, our recognizer using 5 emitting state HMM has better recognition accuracy than 3 emitting state HMM.

The number of words which are found in a sentence has an influence on the recognizer performance. We found that the test data with less number of words within sentences has better recognized by the recognizer than long sentence.

Spontaneous speech recognizer for Amharic developed using 90 minute speech database in a judicial domain contains 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 [28]. We found 41.60% word accuracy with our recognizer, and it is less than the result found by Bantegize [28], from this result we can see that, the recognizer developed using a data from the same domain for training and testing performs better than a data which is not from specified domain. Because the data which is from the same domain has many things in common and there is more probability of words repetition. But our data which is collected from web is very sparse.

The size of data we have used is very low in size if compared with another databases used for other language including a large-scale spontaneous speech database “Corpus of Spontaneous Japanese (CSJ)”. Using CSJ Training data of 510 hours long and around 6.84 M words for language modeling, best Word Error Rate of 25.3% obtained [27]. In addition to data size and

domain, the effect of language model on the speech recognizer can be observed from the works in Bantegize [28] and Furui [27]. Both of them used trigram language model while we have used bigram for this study due to our data size and tool restriction.

The result we have come across during our recognizer test was 41.60% of word accuracy for test data from speakers those are involved in training, 39.86% for test data from both speakers those involved in training and do not involved in training and 23.25% for test data from speakers those do not involved in training. Even though the recognizer has developed under limited resources and insufficient size of data we can say that the result found for spontaneous speech was promising. Although we did up to this figure of accuracy with the time frame and data we have in hand. Whether by adding data size or by using the same data and refining some parameters concerning with language model and dis-fluencies, still it is possible to improve this accuracy.

The accuracy of our ASR developed using spontaneous speech is less than the accuracy of Amharic ASR developed by different researchers Solomon [6], Kinfe [11], and others using read speech. One of the factors which make our recognizer less accurate is data size and the nature of the speech data (conversational). [35] States that, Study of spontaneous speech requires larger amounts of data than in the study of read speech for at least two reasons.

For one thing, spontaneous speech is inherently more variable than read speech. Different groups of speakers have different manners of speaking depending on social attributes such as age, gender, education, profession, and so on. Moreover, one speaker could speak in quite different speaking styles according to social and personal conditions. This inter- and intra-speaker variability requires analyses based on a large data size.

For another thing, a speech sample cannot be called spontaneous if its linguistic message is completely pre-fixed. Spontaneous speech should be constructed by speakers on site. This means that there is not much room for researchers to make experimental design to reduce the time and cost of data collection. Accordingly, a large scale corpus of spontaneous speech is required as well as necessary for understanding the linguistic nature of spontaneous speech in order to make

an advance in the development of speech and natural language processing technologies for application to real human speech.

5.2 Recommendation

From the experimental results and what has been learned throughout the course of this research, we would like to forward our recommendations and future works with the following paragraphs.

There is no spontaneous speech corpus prepared previously which we can use for this study, therefore we have tried to prepare it from the scratch. Although we have prepared spontaneous speech data to use for this study, still it is not sufficient therefore it is the future task for researchers which work on this area. Increasing the size of data has a lot of advantages like improving recognizer accuracy and even to ensure the applicability of ASR. Since it is difficult to use ASR with low accuracy in a real world, we have to work on increment of data size and recognizer accuracy improvement.

In this study we did our experiment using bigram language model. In addition to increasing data size, depending on the size of data in hand doing a lot with language model like increasing N-gram into trigram and other N-grams improves the recognizer accuracy.

The speech data which is used for this study is sparse and diversified and it is from different speakers. Consequently, in order to handle the variability among speakers, the need for investigating the speaker adaption is one of the issues to be handled for the recognizer development; therefore the need for conducting research in this area is also one of the important aspects to be considered in order to improve the accuracy of the recognizer.

One of the difficult feature of spontaneous speech is non-speech events (dis-fluencies), therefore handling none speech events has a big role in recognizer's performance. We have dealt with some issues to handle them in general but still it is possible to handle them further by applying different techniques. May be one of the task one can do in order to handle these non-speech events effect is including or removing from acoustic, lexical and language model, by studying their nature in detail and separately.

Chapter 5: Conclusion and Recommendation

We have used canonical pronunciation dictionary, but there are the same words that were uttered differently by different speakers, dialect variation. It is one of the reasons which decrease the performance of the recognizer. Therefore it is better to use alternative pronunciation dictionary in order to improve recognizer performance.

We did manually the transcription of collected speech data. Therefore there might be some errors during transcription and it is tedious work. Developing a tool which can be applied for this purpose and using it for automatic transcription, decreases the time and efforts that can be wasted for transcription.

References

[1] Jurafsky D., Martin J.H, Speech and Language Processing, An Introduction to

Natural Language Processing, Computational Linguistics and Speech Recognition, Pearson Prentice Hall, Upper Saddle River, N.J, 2009

[2] Gales M. and Young S, The application of Hidden Markov Models in speech recognition,

Cambrige University, Cambrige, uk, 2008

[3] Markowitz, Judith A., Using Speech Recognition, Upper Saddle River, New Jersey,

Prentice Hall, Inc, 1996

[4]Solomon T, Martha Y, Wolfgang M, Amharic Speech Recognition: past, present

And Future, (ed) SveinEge, Harald Aspen, BirhanuTeferra and Shiferaw Bekele, In: Proceedings of the 16th International Conference of Ethiopian Studies, Trondheim, pp 1391-1401, 2009

[5]Burileanu C., spontaneous speech recognition for Romanian in spoken Dialogue

Systems, proceedings of the Romanian academy, Series A, The publishing house of Romanian Academy, Vol 1, pp. 83-91, 2010

[6]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

[7] Rabiner, L.R.,Juang, B-H, Fundamentals of Speech Recognition, New Jersey:

Prentice-Hall, 1993

[8]André Gustavo Adami, Automatic Speech Recognition: From the Beginning to the

Portuguese Language, Brasil,(cf)

[9]Therese S.S. and Lingam Ch., Review of Feature Extraction Techniques in

Automatic Speech Recognition, International Journal of Scientific Engineering and Technology,Volume No.2, Issue No.6, pp : 479-484, 2013

[10]Solomon Berhanu, Isolated Amharic Consonant-Vowel (CV) Syllable Recognition,

An Experiment Using the HMM, M.Sc. Thesis, Addis Ababa University, Addis Ababa, Ethiopia, 2001

[11]Kinfe Tadesse, Sub-word based Amharic speech recognizer: An experiment using

Hidden Markov Model (HMM), MSc Thesis, School of Information Studies for Africa, Addis Ababa University, Ethiopia, 2002

[12]Martha Yifru, Application of Amharic speech recognition system to command and

Control computer: an experiment with Microsoft word, M. Sc. Thesis, Addis Ababa University, Addis Ababa, Ethiopia, 2003

[13]Zegaye Seifu, HMM based large vocabulary, speaker in-dependent, continuous

Amharic speech recognizer, M.Sc Thesis, School of Information Studies for Africa, Addis Ababa University, Ethiopia, 2003

[14]Yitagesu B. Gebramedhin, et al, A new approach to develop syllable based,

Continuous Amharic speech recognizer, euro2013, Zagreb, Croatia,ppt 1684-1689, 2013

[15]Hussien Seid, A Speaker Independent Continuous Amharic Speech Recognizer: an

Experiment using hybrid (HMM-ANN), M. Sc. Thesis, Addis Ababa University, Addis Ababa, Ethiopia, 2004

[16]Lee, Kai-Fu, Context-Dependent Phonetic Hidden Markov Models for Speaker

Independent Continuous Speech Recognition, In Reading in Speech Recognition edited By Alex Waibel and Kai-Fu Lee. pp. 347-365. California: Morgan Kuafman, 1990

[17]Anusuya M.A., Katti S.K., Speech Recognition by Machine: A Review, International Journal of Computer Science and Information Security, College of Engineering Mysore, India, Vol. 6, No. 3 pp 181-205, 2009

[18] Gauvain J. and Lamel L, Large vocabulary speech recognition based on statistical Methods, LIMSI,France, 2003

[19] Hermansky H., Perceptual linear predictive (PLP) analysis for speech, The Journal Of The Acoustical Society of America 87, pp 1738-1752, 1990

[20]Avendaño, C., Deng L., Hermansky H., Gold, B., the Analysis and Representation of Speech, Speech Processing in the Auditory System, pp 63-100, 2004

[21] Wiggers Ir. P. and Rothkrantz L.J.M, Automatic speech recognition using HMM, Data and Knowledge Systems Group, 2003

[22]Molalgne Girmaw, An automatic speech recognition system for Amharic, M.Sc. Thesis, Dept. of Signals, Sensors and Systems, Royal Institute of Technology, Stockholm, Sweden, 2004

[23] Duchateau J., Handling Disfluencies in Spontaneous Language Models, ESAT – PSI Speech Group, K.U. Leuven

[24]Jon P. Nedel, Rita Singh, Richard M. Stern., Automatic sub word unit refinement for spontaneous speech recognition via phone splitting, Carnegie Mellon University, Pittsburgh, 2007

[25]Young S., The HTK Book, Cambridge University Engineering Department, Available from: http://htk.eng.cam.ac.uk , 2009, accessed on 10/01/2014

[26] Furui S., Recent Advances in Spontaneous Speech Recognition and Understanding,

Tokyo Institute of Technology, Department of Computer Science, Japan, 2007

[27] Furui, S., recent progress in corpus-based spontaneous speech recognition, IEICE Trans. Inf. & Syst., E88-D, 3, pp. 366-375, 2005

[28]Bantegize Addis, BRANA:Application of Amharic speech recognition system for Dictation in judicial domain, MScThesis, Gondar University, Gondar, Ethiopia, 2015

[29]Baye Yimam, የአማርኛሰዋሰዉ, Addis Ababa, ት.መ.ማ.ማ. ድ, ነ986

[30]Baye Yimam and TEAM 503 students, የአማርኛ ሰዋሰዉ, Ethiopian Journal of LanguagesAnd Literature 7 pp 1-32, 1997

[31]Muhie Yimam S., "TETEYEQ (ተጠየቅ): Amharic question answering for factoid questions", M.Sc. Thesis, Addis Ababa University, Ethiopia, 2009

[32]Tadesse Beyene, The Ethiopian Writing System, Paper presented at the 12th International Conference of Ethiopian Studies, Michigan State University, 1994

[33] Butzberger J., Murveit H., Shriberg E., Price P.,: Spontaneous Speech Effects in Large Vocabulary Speech Recognition Applications: SRI International Speech Research and Technology Program, Menlo Park, CA 94025

[34]Furui S., Spontaneous speech corpus of Japanese, National Language Research Institute, Tokyo Institute of Technology, Tokyo Japan, 2002

[35]Furui S., Toward Spontaneous Speech Recognition and Understanding, In: Wu, C., Bing Huang, J. (eds.), Pattern Recognition in Speech and Language Processing, CRC Press, Inc., Boca Raton, pp 191-227, 2003

[36]Bender L.M. and Ferguson C., The Ethiopian Writing System, in Bender, M. et al (eds.), Languages in Ethiopia, London, Oxford University Press, 1976

[37] Hayward, Katrina and Richard J. H, Amharic, In Handbook of the International Phonetic Association: A guide to the use of the International Phonetic Alphabet, Cambridge: the University Press, 1999

[38]Mesfin Birile, synthetic speech trained -large vocabulary Amharic Speech Recognition, M.Sc. Thesis, Addis Ababa University, Addis Ababa, 2008

[39]Rangarajan .V and Narayanan S, Analysis of dis-fluent repetitions in spontaneous Speech Recognition, Speech Analysis and Interpretation Laboratory, University of Southern California, California, 2008

[40] Shrawankar U., Thakare V., Techniques for feature extraction in speech recognition System: a Comparative study, SGB Amravati University, Amravati

Appendix

Appendix A:

Sample Pronunciation Dictionary

BR	[BR]	BR sp
CEbITEwI	[CEbITEwI]	C E b I T E w I sp
CElEmEmIbINI	[CElEmEmIbINI]	C E l E m E m I b I N I sp
CEle	[CEle]	C E l e sp
CEmErIkI	[CEmErIkI]	C E m E r I k I sp
…	…	…
~yEmI	[yEmI]	y E m I sp
~yEmilEwI	[yEmilEwI]	y E m i l E w I sp

Sample Pronunciation Dictionary for HDecode

BR	[BR]	BR
CEbITEwI	[CEbITEwI]	C E b I T E w I
CElEmEmIbINI	[CElEmEmIbINI]	C E l E m E m I b I N I
CEle	[CEle]	C E l e
CEmErIkI	[CEmErIkI]	C E m E r I k I
…	…	…
~yEmI	[yEmI]	y E m I
~yEmilEwI	[yEmilEwI]	y E m i l E w I

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

HVite tool config file

TARGETKIND = MFCC_0_D_A

TARGETRATE = 100000.0

SAVECOMPRESSED= TRUE

SAVEWITHCRC = TRUE

WINDOWSIZE = 250000.0

USEHAMMING = TRUE

PREEMCOEF = 0.97

NUMCHANS = 26

CEPLIFTER = 22

NUMCEPS = 12

ENORMALISE = FALSE

SOURCEFORMAT = HTK

USESILDET = TRUE

MEASURESIL = FALSE

OUTSILWARN = TRUE

MICIN = TRUE

SOURCERATE = 625

FORCECXTEXP = TRUE

ALLOWXWRDEXP = FALSE

Appendix D:

‘Proto’ file for 3 emitting state HMM

~o <VecSize> 39 <MFCC_0_D_A>

~h "proto"

<BeginHMM>

<NumStates> 5

<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 …

<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> 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 …

<TransP> 5

0.0 1.0 0.0 0.0 0.0

0.0 0.6 0.4 0.0 0.0

0.0 0.0 0.6 0.4 0.0

0.0 0.0 0.0 0.7 0.3

0.0 0.0 0.0 0.0 0.0

<EndHMM>

Appendix E:

'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 …

<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> 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

<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…

<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…

<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" }
```

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" }

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-*" }

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-*" }

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-*" }

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]}

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]}

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]}

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]}

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]}

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]}

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]}

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]}

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"

Declaration

This thesis is my original work and has not been submitted as a partial requirement for a degree in any other university and that all source of materials used for the thesis have been properly acknowledged.

_____

Adugna Deksiso

March, 2015

The thesis has been submitted for examination with my approval as university advisor.

_____

Martha Yifiru (Ph.D)