Developing Tigrinya Speech Recognizer Using Amharic and Tigrinya Data

Dionasios Deressa

A thesis Submitted to

the Department of Linguistics

Presented in Partial Fulfillment of the Requirements for the Degree of Master of Computational Linguistics

Addis Ababa University

Addis Ababa, Ethiopia

March 2015

Addis Ababa University

School of Graduate Studies

This is to certify that the thesis prepared by Dionasios Deressa, entitled: Developing Tigrinya Speech Recognizer Using Amharic and Tigrinya Data and submitted in fulfillment of the requirements for the Degree of Master of Computational Linguistics compiles with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the Examining Committee:

Examiner _____Signature _____Date_____

Examiner _____Signature _____Date_____

Advisor _____Signature _____Date_____

Advisor _____Signature _____Date_____

_____

Chair of Department or Graduate Program Coordinator

Abstract

This study has introduced the design of a Hidden Markov Model based LVCSR system in a new target language based on a different source language and without the need of a large speech databases on the target language. The Tigrinya LVCSR was developed using an Amharic Corpus consisting of 10,850 sentences and a limited Tigrinya data containing 600 sentences to train the acoustic models. The study was conducted based on the knowledge based approach taking the assumption that the articulatory representations of phonemes are similar across the Tigrinya and Amharic languages with the exception of 2 phonemes unique to Tigrinya and using all the phonemes as acoustic units.

A total of six experiments were performed using different parameters each one done in an effort of increasing the performance of the recognizer. Out of the five experiments, the best result obtained with the experiment that is done by training the seed model with the 10,850 Amharic sentences up to the 8^{th} iteration and using the 600 Tigrinya sentence starting from the 8^{th} iteration of the training process. The experimental result showed percentage of correctly recognized words of 88.33% with an accuracy of 73.43 %.

The baseline Tigrinya recognizer which was trained using only the 600 Tigrinya data resulted in correctly recognized words of 80.80% and an accuracy of 67.39 % on the tri-phone model with 12 Gaussian mixtures. Comparing this result with the best result obtained in the experiment showed that an increase of about 8% was achieved in terms of correctly recognized words and of about 6% in terms of accuracy. This has proven that the use of Amharic data with limited Tigrinya data for training a Tigrinya recognizer does result in significant performance increase and that it is a promising future research direction given that different methods are applied to further achieve better results. As this is the first attempt other phone mapping techniques and approaches such as the data driven approach can also be tried for performance improvement purpose.

Acknowledgment

First and foremost I would like to thank God for making all this possible. My deepest gratitude also goes to my advisor Dr. Martha Yifiru and Dr. Mulugeta Seyoum for the useful guidance and support they offered throughout the process of conducting this work.

I would also like to thank Dr. Solomon Teferra and Ato Hafte Abera for allowing me to use their speech corpus, without which this research would have been impossible. Finally I would like to thank my family and friends for their constant support and love.

Table of Contents

List of Acronyms

ASR	Automatic Speech Recognition
CFG	Context Free Grammar
CV	Consonant Vowel
DNA	Deoxyribonucleic Acid
DFT	Discrete Fourier Transform
EM	Expectation Maximization
HMM	Hidden Markov Model
HTK	Hidden Markov Model Toolkit
IPA	International Phonetic Alphabet
LM	Language Model
LP	Linear Prediction
LPC	Linear Prediction Coding
LVCSR	Large Vocabulary Continuous Speech Recognition
MFCC	Mel Frequency Cepstral Coefficient
MLE	Maximum Likelihood Estimation
MLF	Master Label File
MMF	Master Macro File
NLP	Natural Language Processing
PLP	Perceptual Linear Prediction
SLF	Standard Lattice Format
SLM	Statistical Language Model
SRILM	Stanford Research Institute Language Modeling
WER	Word Error Rate

List of Figures

List of Tables

CHAPTER ONE

Introduction

1.1 Background

Speech is the primary means of communication between people. For reasons ranging from technological curiosity about the mechanisms for mechanical realization of human speech capabilities, to the desire of automating simple tasks inherently requiring human-machine interactions, research in automatic speech recognition by machine has attracted a great deal of attention over the past six decades (Das, 2012:2071).

"The Speech Recognition Technology also known as Automatic Speech Recognition (ASR) or computer speech recognition is the process of converting a speech signal to a sequence of words, by a means of an algorithm implemented as a computer program" (Anusuya & Katti, 2009:181).

This technology has made it possible for computers to follow human voice commands and understand human languages with the main aim of developing techniques and systems for speech input to machines (Anusuya & Katti, 2009). Based on major advances in statistical modeling of speech, automatic speech recognition systems today find widespread application in tasks that require human machine interface, such as automatic call processing in telephone networks, and query based information systems that provide updated travel information, stock price quotations, weather reports, data entry, voice dictation, railway reservations etc. (Das, 2012:2073).

The design of these automatic speech recognition technologies mainly incorporates three major components, an acoustic model, a lexical model and a language model (Rahul , Venkatesh, Anumanchipalli, & Joshi, 2005). The acoustic model attempts to model the sound units of a language based on speech features extracted from an audio signal. The sound units of a language representing this model could be phones, sub-phones or

syllables. The second component, which is the lexical model, provides a list of transcription of words based on the type of sound unit used which is needed for training as well as recognition purposes. The last and crucial component, the language model represents the sequence of words in the context of the task being performed by the speech recognizer. The importance of this model lies in its ability to reduce the search space of a sequence of words hypothesized by the recognizer.

Hidden Markov Model (HMM) based acoustic modeling is one most known and commonly used statistical approach. Other acoustic models include segmental models, super-segmental models (including hidden dynamic models), neural networks, maximum entropy model, and (hidden) conditional random fields (Young & Gales, 2008).

HMM based acoustic models for a certain language require training using speech corpus from that language. The speech corpus would incorporate an audio data along with the transcriptions of the data in that language. Research in ASR is now focusing on systems that incorporate three features: large vocabularies, continuous speech capabilities, and speaker independence (Das, 2012:2072). The size of the speech corpus would depend on the ASR being large, medium or small vocabulary. The former one requires hundreds of hours of training data to achieve state of the art performance.

Researchers have long been striving to produce a practical cross-language solution to the problem of data unavailability. Efforts have largely been focused on borrowing resources from resource-rich languages to build acoustic models for a resource-poor language (Liu & Melnar, 2006).

In this regard, representative approaches are roughly divided into two categories, linguistic and acoustic (Liu & Melnar, 2006). In the former approach, phonetic or phonological knowledge is used to select phonemes from the source languages as substitutes for the target-language phonemes. The selection procedure is normally determined either through IPA symbol commonality or through a more in-depth consideration of phonological factors by a language expert.

On the other hand, the acoustic approach is data-driven, where some native data from the target language is required. The native data is either employed to further improve the performance of the models built with the linguistic approach, or it is used to train raw models, which in turn are used to locate the best candidate models in the source languages through acoustic distance measurement (Naveen & Shrikanth, 2003). Generally, data-driven approaches have yielded cross-language models with superior performance relative to approaches using only knowledge-based phoneme mappings (Juang & Rabiner, 2004).

1.2 Statement of the Problem

Speech corpus based technology has been widely used in people's lives, although it is still a strange concept for many people. Speech corpus, the collection of speech signal, its transcription, metadata and documents, is the basis for both analyzing the characteristics of speech signal and developing speech synthesis and recognition systems (Ai-jun & Zhi-gang, 2006).

Development of robust spoken language technology ideally relies on the availability of large amounts of speech data preferably in the target language. However, more often than not, speech developers need to cope with very little or no data, typically obtained from a different target domain (Naveen & Shrikanth, 2003).

The past few years has seen tremendous success for automatic speech recognition (ASR) in several different resource rich languages. Most of the approaches used in ASR development are statistical, which mainly rely on the availability of large amounts of speech and text data. Thus, due to this reason the rapid transfer of ASR technologies to new languages is hampered by the non-availability of sufficient data (Naveen & Shrikanth, 2003).

Several researchers have employed different methods to tackle this problem. Tanja and Alex (2001) developed ASR for a new target language using speech data from varied

source languages, but only limited data from the target language. The research was based on the assumption that the articulatory representations of phonemes are so similar across languages that phonemes can be considered as units which are independent from the underlying language and thus, unified into one global set.

The adaptation of ASR technology in Ethiopian languages also suffers from lack of sufficient data. Little has been done in the preparation of speech corpus in the local languages. The only available speech corpus is an Amharic speech corpus developed by Solomon, Wolfgang, and Bairu (2005) containing 10,850 sentences. For other local languages, including Tigrinya, there is no standard corpus except small size data prepared by students for academic research purposes.

Hafte (2009) developed a speaker independent continuous speech recognizer for Tigrinya language using 300 sentences recorded from 12 people that are native speakers of the language. A final recommendation made by the researcher was the need for a large vocabulary speech corpus to develop a robust large vocabulary speech recognizer. However, development of a large vocabulary speech corpus is time, labor and resource intensive and cannot be attempted within a short period of time. The use of an already developed speech corpus of other languages is another alternative for the development of speech recognition systems with limited language resources to achieve performance improvement.

According to Shimelis (2014) the articulatory representations of phonemes are similar across the Tigrinya and Amharic languages. Tigrinya has 39 phonemes among which 2 are unique to the language and not found in Amharic.[1] Taking their similarity as a basis, the purpose of this study is to investigate the possibility of developing a Tigrinya speech recognizer, using a large vocabulary Amharic read speech corpus developed by Solomon et al. (2005) and a limited Tigrinya data. This method of developing a recognizer hasn't been tried for Ethiopian languages therefore making this study the first attempt.

[1] Unique phonemes refer to the sounds being unique to Tigrinya as compared to the Amharic language or the phonemes not being found in the Amharic language.

To this end, this study explores and answers the following research questions,

- Is it possible to develop a large vocabulary continuous Tigrinya speech recognizer using Amharic training data set?
- Does the approach followed in building a LVCSR system for a new target language (Tigrinya) using speech data from source language (Amharic) result in better performance than speech recognizer trained with insufficient Tigrinya data?

1.3 Objective of the Study

1.3.1 General Objective

The general objective of the study is to investigate the possibility of developing a Large Vocabulary Continious Tigrinya Speech Recognition system using an Amharic Corpus and limited Tigrinya data to train the acoustic model.

1.3.2 Specific Objectives

In order to achieve the above general objective, the following specific objectives are set.

- To conduct a literature review on the phonetics and phonology of the Tigrinya and Amharic lanuages, and understand the various approaches used in ASR development, and the HTK tool.
- To prepare limited Tigrinya speech corpus for adaptive acoustic model training and testing the recognizer.
- To collect Tigrinya text and construct a Tigrinya language model.
- To design and develop a prototype Tigrinya speech recognizer.
- To test and analyze the performance of the Tigrinya recognizer.
- To draw conclusions and provide recommendations for further study.

1.4 Scope and Limitation of the Study

This study covers the investigation and the possibility of developing an HMM based speaker independent LVCSR for Tigrinya using phones and tri-phones as acoustic units.[2] Amharic data containing 10,500 sentences and Tigrinya data containing 600 sentences was used to develop the recognizer using the knowledge based approach.

The study is limited to training of phone and tri-phone models for the Tigrinya recognizer and does not handle the **detailed** examination of the left and right context of the Tigrinya phones for phone clustering and parameter tying.

1.5 Methodology

Methodology refers to the principles, procedures, and practices that govern research. In order to achieve the objectives stated in section 1.3, the research employed an experimental quantitative research methodology where there is much greater control over the research environment. Different methods for conceptual understanding, data set preparation, designing the system and evaluation of the recognizer were applied.

1.5.1 Literature Review

A Literature review was conducted to summarize and compare some of the well-known approaches used in Speech recognition for a target language using different source languages. Previous works done on different types of speech recognition systems were also reviewed. Various resources, such as books, journal articles and conference papers were reviewed for the purpose of conducting the research and further understanding the Tigrinya and Amharic language.

[2] Phones are the smallest sound units of which words are composed. Phone models are highly trainable. Tri-phones or context dependent phones model a phone in its left and right context and thus are powerful models as they are able to capture the co-articulation effect (Lee, 1989).

1.5.2 Data Source

The principal drawback in building speech recognizers is their dependency on large amount of speech databases to train the models. The statistical methods applied to speech and language modeling not only require hours of recorded and transcribed speech, but also pronunciation dictionaries and large text corpora.

The Amharic speech corpus used for training the Tigrinya recognizer is a read speech corpus prepared by Solomon et al. (2005). It consists of a training set, development test sets and evaluation test sets out of which only the training set was used. The Amharic corpus contains 20 hours of training speech collected from 100 speakers who read a total of 10,850 sentences or 28,666 tokens.[3] Eighty of the training speakers are from the Addis Ababa dialect while the other twenty are from other four dialects.

In addition to the Amharic corpus, 300 Tigrinya sentences previously recorded by Hafte (2009) were also used. However the data was labeled data and thus could not be used for testing and thus was used for adapting the models. The coverage of the unique Tigrinya phones was also small in this data. Due to this reason, Tigrinya speech data was also collected and processed for further adaptation and testing purposes. The total amount of Tigrinya data collected consisted of a total of 600 sentences or 2 hours of read speech data along with its set of transcription text.

The text data from the speech transcriptions of the train and test data was used for the preparation of the train and test dictionary respectively.

As there was no Tigrinya text data previously collected for language model development, Tigrinya text data from various Internet sources like newspapers, websites, articles, e-books was collected through a tedious and time consuming process. It consisted of 50,000 sentences or 1,341,578 tokens.

[3] Token refers to words or symbols.

To develop the training as well as test pronunciation dictionaries, a wordlist consisting of the unique words (distinct words or single instance of the word) found in the training and test transcription data were used respectively.

1.5.3 Development Tool

The Hidden Markov Model Toolkit (HTK) was used as a development tool. HTK is a portable toolkit for building and manipulating hidden Markov models (HMM) (Young et al., 2009). It is primarily used for speech recognition research, although it has been and still is being used for numerous other applications including, research in speech synthesis, character recognition and DNA sequencing (Woodland, 2014). The tool provides sophisticated facilities for speech analysis, HMM training, testing and results analysis. The software also supports HMMs using continuous density mixture Gaussians, discrete distributions and can be used to build complex HMM systems.

Language model development was done using SRI Language Modeling toolkit (SRILM), which is a toolkit for building and applying different types of statistical language models (Andreas, 2002).

1.5.4 Evaluation Procedure

Accuracy is one of the metric used to evaluate a speech recognizer. It is measured with the percentage of correctly recognized words and Word Error Rate (WER). Deletion, substitution and insertion are the three types of errors usually observed in large vocabulary continuous speech recognizers (Adami, 2010). Deletion error occurs when a word is omitted from the recognized sentences whereas substitution results when another word is recognized in place of the correct one. An insertion error arises when an extra word is added.

$$Correct\ Word\ \ Rate = \frac{N - D - S}{N} * 100$$

$$Accuracy = \frac{N - D - S - I}{N} * 100$$

Where I denote Insertions, D denotes Deletions, S substitutions and N the total number of words in the recognized sentences.

1.6 Significance of the Research

The proposed research will have practical as well as methodological significance. Designing a LVCSR for Tigrinya language, enables the speakers of the language to make use of the different application areas that speech recognition systems have. The approach and methodologies followed towards the development of the recognizer using limited amount of data can be used as a basis for the portability of recognizers in other similar local languages.

1.7 Organization of the Thesis

The research paper is organized into five chapters. Chapter one introduces the brief concepts on ASR, its status and gives the basis for conducting the research work. General as well as specific objectives, justification of the study and methods employed to conduct the study are also discussed here.

Chapter two gives a detailed explanation on Speech Recognition and its component, on the approaches and algorithms used for developing the recognizer and the speech signal processing steps involved. Previous works done on similar research areas that shed light on the status of local and international works are given here.

Chapter three presents the phonetics and phonology Tigrinya and Amharic language, the writing system and a comparison of the phonetic system of the two languages.

Chapter four deals with a detailed explanation of the steps taken to construct a LVCSR for Tigrinya using a different source language, Amharic for the purpose of training the recognizer. The data set used, the processes executed for data collection and preprocessing, training the acoustic and language model as well as the steps accomplished at the recognition stages are described here in detail. The experimentations done by changing and tuning different parameters in order to achieve better results are also described.

Chapter five gives conclusions drawn based up on the experiments conducted and presents the achievements of the research, it shortcomings and draws recommendations from them.

CHAPTER TWO

Literature Review

Automatic Speech Recognition (ASR)

2.1 Introduction

Automatic Speech Recognition (ASR) is the process of converting a speech signal or an observation sequence to a string of words (Anusuya & Katti, 2009). It is a system that automatically transcribes speech into text. ASR can also be defined as a mechanism that is capable of decoding an acoustic signal produced by a human speaker into a sequence of linguistic units contained in the message that the speaker wants to communicate (Peinado & Segura, 2006).

2.2 Speech Recognition Types

Speech recognition systems can be classified in to several different classes by describing what types of utterances they have the ability to recognize. Figure 2.1 shows the different types of speech recognition systems based on the speech mode, speaker mode, vocabulary size and speaking style.

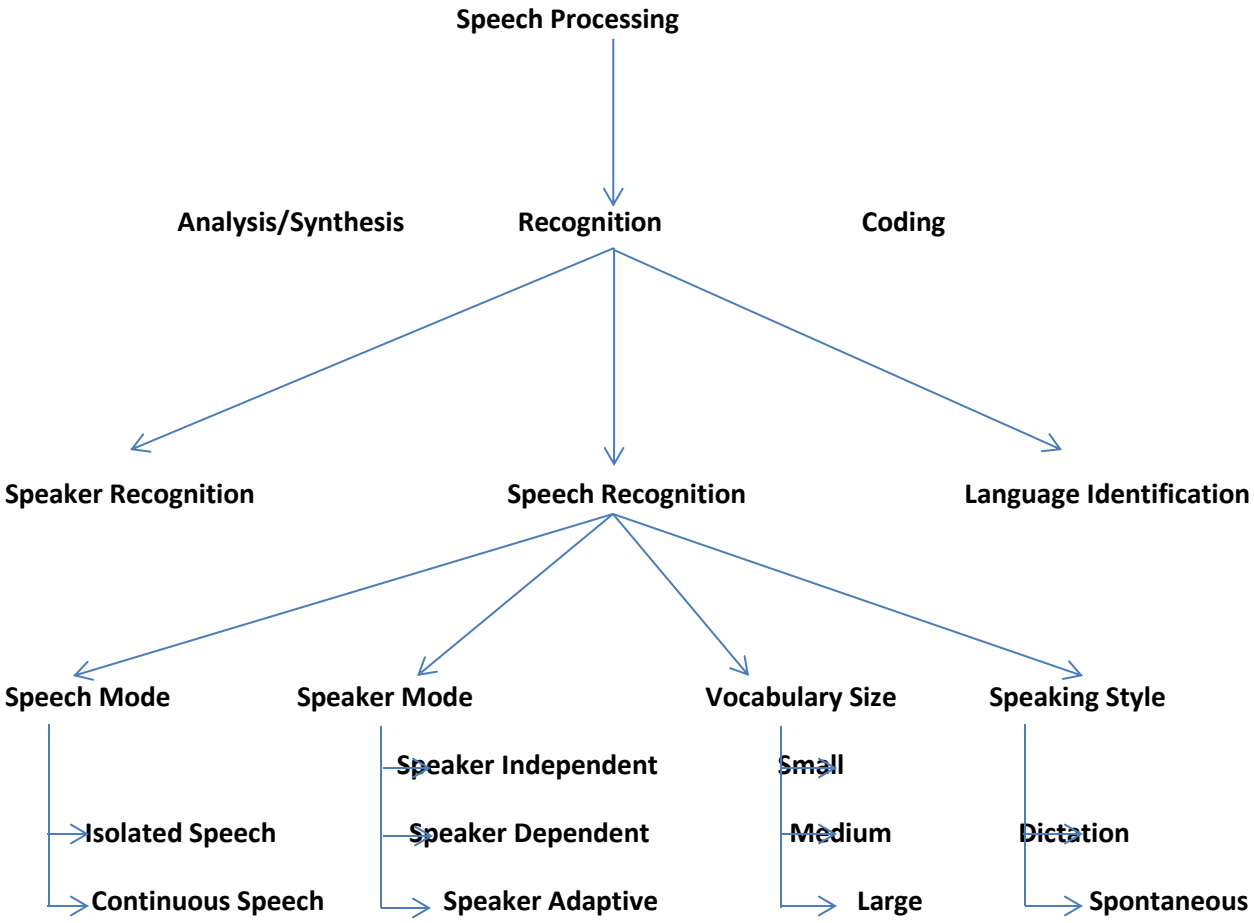

Figure 2. 1 Categorization of ASR (Anusuya & Katti, 2009).

The speech mode is either isolated speech or continous speech. Isolated word recognizers usually require each utterance to have silence or lack of an audio signal on both sides of the sample window (Pires, 2007). It accepts single words or single utterance at a time.

On the other hand, continuous speech recognizers allow users to speak almost naturally, while the computer determines the content. Recognizers with continuous speech capabilities are some of the most difficult to create because they utilize special methods to determine utterance boundaries (Anusuya & Katti, 2009).

Speech recognition can also be classified into dictation/read and spontaneous. Spontaneous speech can be thought of as speech that is natural sounding and not rehearsed or read. An ASR system with spontaneous speech ability should be able to

handle a variety of natural speech features such as words being run together, "ums" and "ahs", and even slight stutters (Singh et al., 2012). On the other hand, the read speech or dictation type of ASR is where the speech is read from a prompt and not impulsive.

In addition to the classification based on the type of utterances, speech recognizers can also be classified based on speaker modes as speaker dependent or independent and based up on the vocabulary size as small, medium and large.

Speaker dependent models: Speaker dependent systems are designed for a specific speaker. They are generally more accurate for the particular speaker, but much less accurate for other speakers. This systems are usually easier to develop, cheaper and more accurate, but not as flexible as speaker adaptive or speaker independent systems (Adami, 2010).

Speaker independent models: Speaker independent systems are designed for a variety of speakers. It recognizes the speech patterns of a large group of people (Das, 2012:2072). This system is most difficult to develop, most expensive and offers less accuracy than speaker dependent systems.

The vocabulary size of speech recognition systems affects the complexity, processing requirements and the accuracy of ASR systems (Adami, 2010). Some applications only require a few words, others require very large dictionaries. Small vocabulary ASRs are able to recognize tens of words, Medium vocabulary hundreds of words, large vocabulary thousands of words, and very-large vocabulary tens of thousands of words (Anusuya & Katti, 2009).

Apart from the above characteristics, the environment variability, channel variability, speaker style, gender, age, speed of speech also make the ASR system more complex (Das, 2012). But an efficient ASR system must cope with the variability in the signal.

2.3 Components of Speech Recognition System

Automatic Speech Recognizers have different components that participate in the recognition process. These participating components are the acoustic model, language model and the lexical model (pronunciation dictionary) (Jurafsky & Martin, 2007). As indicated in Figure 2.2, whenever a speech signal is delivered to the system, it will be processed by these different components to produce its equivalent string of output word sequences.

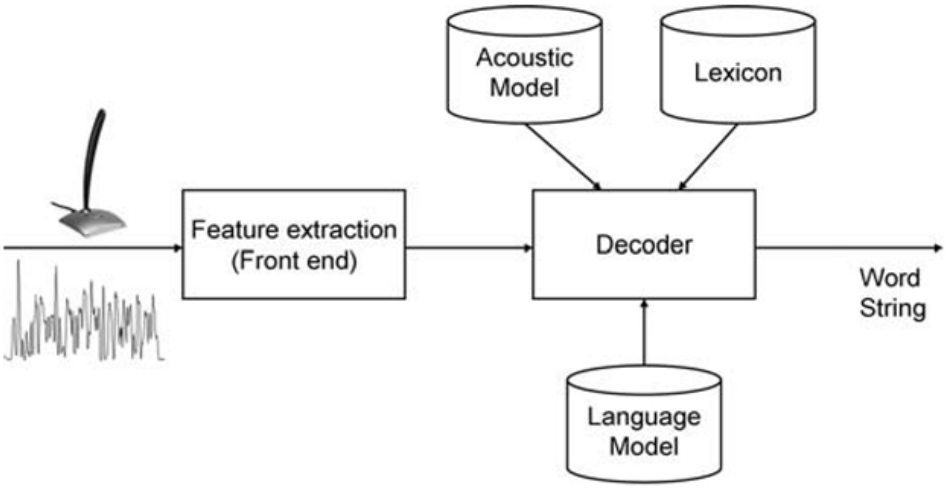

Figure 2. 2 Components of a Speech Recognizer.[4]

For a given sequence of acoustic observation $O = o_1 o_2 \ldots . o_n$, the goal of speech recognition is to find out the corresponding word sequence $\hat{W} = w_1 w_2 \ldots w_m$ that has the maximum a posteriori probability (MAP) P (**W** | **O**):

$$\hat{W} = \underset{w}{\operatorname{argmax}}\, P(W|O) = \underset{w}{\operatorname{argmax}}\, \frac{P(W)P(O|W)}{P(O)} \qquad 2.1$$

Where *P*(*W*), the **prior probability**, is computed by the **language model** and *P*(*O*|*W*), the **observation likelihood**, is computed by the **acoustic model** (Jurafsky & Martin, 2007).

[4] http://what-when-how.com 15/3/2015

2.3.1 Speech Signal Processing

Preparation as well as processing of speech utterances to make them ready for training and recognition requires a number of speech signal processing stages. The first one is to convert the analog representations first air pressure, and then analog electric signals in a microphone, into a digital signal (Kesarkar, 2003:1). This process involves two steps such as sampling and quantization process.

A signal is first sampled by measuring its amplitude at a particular time, thus the sampling rate is the number of samples or amplitude measurements taken per second. The sampling rate is represented using the Hertz. Most information in human speech is in frequencies below 10,000 Hz which means a 20,000 Hz sampling rate would be necessary for a complete accuracy. An 8,000 Hz sampling rate requires 8000 amplitude measurements for each second of speech, and so it is important to store the amplitude measurement efficiently (Jurafsky & Martin, 2007).

The amplitude measurements are usually stored as integers, either 8-bit values from -128–127 or 16 bit values from -32768–32767. The process of representing real-valued numbers as integers is called quantization because there is a minimum granularity (the quantum size) and all values which are closer together than this quantum size are represented identically. Each sample in the digitized quantized waveform is referred to as $x[n]$, where n is an index over time (Jurafsky & Martin, 2007).

Following the transformation of the analog signal into a digital representation, the speech samples have to be converted into feature vectors. This is achieved by the parameterization process or feature vector extraction.

2.3.2 Feature Extraction

Feature extraction is the process of retaining useful information of the signal while discarding redundant and unwanted information (Singh et al., 2012). Only if the useful speech information can be extracted and represented in a suitable form in the pre-processing part, can the rest of the recognizer work properly. Thus, feature vector extraction stage seeks to provide a compact representation of the speech waveform. Feature vectors are typically computed every 10 ms using an overlapping analysis window of around 25ms (Young & Gales, 2008). Many feature extraction techniques are available, including linear predictive coding, Mel Frequency cepstral coefficient, and perceptually based linear predictive analysis.

Linear Predictive Coding (LPC)

The basic idea behind the linear predictive coding (LPC) analysis is that a speech sample can be approximated as linear combination of past speech samples. By minimizing the sum of the squared differences over a finite interval between the actual speech samples and the linearly predicted ones, a unique set of predictor coefficients is determined. Speech is modeled as the output of linear, time-varying system excited by either quasi-periodic pulse during voiced speech, or random noise during unvoiced speech (Kesarkar, 2003:5).

Mel Frequency Cepstral Coefficients

This analysis technique uses cepstrum with a nonlinear frequency axis following mel scale. For obtaining Mel cestrum the speech waveform is first windowed with analysis window and then its DFT is computed. The magnitude is then weighted by a series of mel filter frequency responses whose center frequencies and bandwidth roughly match those of auditory critical band filters. The use of Mel Frequency Cepstral Coefficients can be considered as one of the standard method for feature extraction. Using about 20 MFCC coefficients is common in ASR, although 10-12 coefficients are often considered to be sufficient for coding speech (Young & Gales, 2008).

Perceptually Based Linear Predictive Analysis (PLP)

The PLP models perceptually motivated auditory spectrum by a low order all pole function, using the autocorrelation LP technique. It involves two major steps, obtaining auditory spectrum and approximating the auditory spectrum by an all pole model. Auditory spectrum is derived from the speech waveform by critical band filtering, equal loudness curve pre-emphasis, and intensity loudness root compression (Kesarkar, 2003). The PLP analysis provides similar results as with LPC analysis but the order of PLP model is half of LP model.

2.3.3 Acoustic Model

An acoustic model in Automatic Speech Recognition is the representation of the relationship between an audio signal and the phonemes or other linguistic units used that make up speech. The model is learned from a set of audio recordings and their corresponding transcriptions created by taking audio recordings of speech, and their text transcriptions, and using different methods or algorithms to create representations of the sounds that make up each word. One such algorithm that is used to train HMM based models is the Baum-Welch training algorithm (Shu-qin, Shao-qian, & Yin-xia, 2012).

2.3.4 Language Models

The goal of language modeling is to characterize, capture and exploit the restrictions imposed on the way in which words can be combined to form sentences and by doing so to describe how words are arranged in a natural language (Jurafsky & Martin, 2007). It is an attempt to capture the inherent regularities (in word sequence) of a natural language making it fundamental to many natural language applications.

Approaches to language modeling are divided in two major groups, deterministic (grammar-based) and stochastic (statistical).” Grammar-based language models are designed by experts on the basis of their knowledge of a language, function of a language model (LM) under development and intuition about the best way to represent linguistic entities and relations in a formal way” (Ilya, 2008:6).

The second technique, the statistical method uses corpus based probabilistic models which are widely applied in natural language processing. Statistical Language models emerge as a result of unsupervised LM estimation on a training corpus. A statistical language model assigns a probability to a sequence of words by means of a probability distribution. It usually starts as a set of void parameters that are estimated in course of observation of language data (Ilya, 2006).

Deterministic or Grammar-Based Language Models

Popular grammar based language models to use in applications are context-free grammars (CFG). A context-free grammar basically consists of a finite set of grammar rules. These grammars consist of sets of terminal symbols, non-terminal symbols and rewriting rules. Terminal symbols are low level units that actually constitute an input string for the grammar to analyze. Non-terminals are higher order constituents that form nodes in the parse-tree for an input sentence (Ilya, 2008:6).

Statistical Language Models (SLM)

The goal of Statistical Language Modeling is to build a language model that can estimate the distribution of natural language as accurate as possible. It is a probability distribution P(s) over strings S that attempts to reflect how frequently a string S occurs as a sentence. By expressing various language phenomena in terms of simple parameters in a statistical model, SLMs provide an easy way to deal with complex natural language in computer (Andreas, 2002). N-grams are examples of statistical language models.

N-Grams

The basic idea behind n-gram models is to consider the structure of a text, corpus, or language as the probability of different words occurring alone or occurring in sequence (Jurafsky & Martin, 2007). The probability of occurrence of string s, p(s) is expressed as:

$$p(s) = p(w_1)p(w_2|w_1)p(w_3|w_1w_2)\dots p(w_1|w_1 \dots w_{1-1}) \qquad 2.2$$

The unigram model treats the words in isolation and determines the frequency of occurrence of each word. In bigram models, the approximation is made that the probability of a word p(w_i) only depends on the identity of the immediately preceding word w_{i-1}; hence p(s) is approximated as:

$$p(s) = p(w_i|w_{i-1}) \quad 2.3$$

For trigram models, the approximation is made that the probability of a word p(w_i) depends on the identity of the immediately preceding word $w_{i-1}$and the word before that w_i ;

$$p(s) = p(w_i|w_{i-1}, w_i) \quad 2.4$$

The parameters in a traditional N-gram model can be estimated with (Maximum Likelihood Estimation) MLE technique:

$$p(w_i|w_{i-1}) = \frac{c(w_{i-1},w_i)}{c(w_{i-1})} \quad 2.5$$

Where $c(w_{i-1}, w_i)$ the number of times w_i is preceded by w_{i-1} and $c(w_{i-1})$ is number of times word w_{i-1} occurs.

Class-based N-gram model

Class-based N-gram models were proposed in order to cope with the data sparseness problem. Instead of dealing with separated words, class-based N-gram estimates parameters for word classes. By clustering words into classes, a class-based N-gram model can reduce the model size significantly with the cost of slightly higher perplexity (Ilya, 2008).

Sequence N-gram model

Sequence N-gram is an attempt to extend N-gram models with variable length sequences. A sequence can be a sequence of word, word class, part-of-speech or whatever a sequence of something that the modeler believes bearing important grammar information (Ilya, 2006).

2.3.5 Lexical Model

The lexical model or pronunciation dictionary specifies the words that may be output by the speech recognizer. It provides a list of transcription of all words based on the type of sound unit used which is needed for training as well as recognition purposes.

2.4 Hidden Markov Model

The Hidden Markov Model (HMM) is a powerful statistical tool for modeling generative sequences that can be characterized by an underlying process generating an observable sequence. The time variances in the spoken language are modeled as Markov process with discrete state spaces. Each state produces speech observations according to the probability distribution characteristics of that state. The speech observations take on a discrete or a continuous value. The states in the model are not directly observable, which is why the model is called Hidden Markov Model (Shu-qin et al., 2012).

HMMs have found applications in many areas interested in signal processing, and in particular speech processing, but have also been applied with success to low level NLP tasks such as part-of-speech tagging, phrase chunking, and extracting target information from documents (Blunsom, 2004).

Since Hidden Markov Models (HMMs) provide a simple and effective framework for modeling time-varying spectral vector sequences. Almost all present day large vocabulary continuous speech recognition (LVCSR) systems are based on HMMs (Young & Gales, 2008).

2.4.1 HMM Architecture

A Markov model is a finite state machine which changes state once every time unit and each time t that a state k is entered, a speech vector o_t is generated from the probability density $b_k(o_t)$ as shown in figure 2.3. The transition from state i to state k is also probabilistic and is governed by the discrete probability a_{ik}. The joint probability that O is generated by the model M moving through the state sequence X is calculated simply as the product of the transition probabilities and the output probabilities (Young et al., 2009).

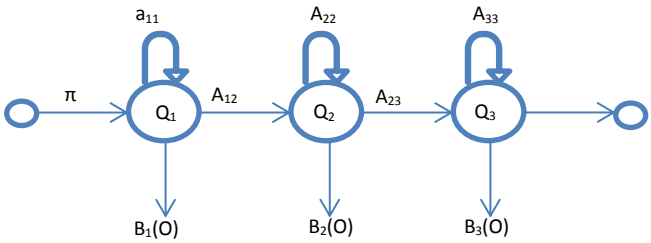

Figure 2. 3 Hidden Markov Model (Jurafsky &Martin, 2007: 325).

As shown in figure 2.3 an HMM model of speech recognition is parameterized by:

$Q = q_1 q_1 \ldots q_N$	A set of **states**
$A = a_{01} a_{02} \ldots a_{n1} \ldots a_{nm}$	A **transition probability matrix** A, each $a_{i\,k}$ representing the probability of going from state i to k.
$B = b_i(o_t)$	A set of **observation likelihoods:**, also called **emission probabilities**, each expressing the probability of a cepstral feature vector (observation o_t) being generated from state i.

2.4.2 HMM Assumption

First-order Hidden Markov Model instantiates the assumptions that the probability of a particular state is dependent only on the previous state (Jurafsky & Martin, 2007). Here it is assumed that state transition probabilities are independent of the actual time at which transitions takes place. Equation 2.6 below shows this assumption:

$$Markov\ Assumption: P\ (q_i|q_1 \ldots\ldots\ldots q_{i-1}) = P(q_i|q_{i-1}) \qquad 2.6$$

The second assumption dictates that the likelihood of generating each observation is dependent only upon the state and is independent of all the other observations (Jurafsky & Martin, 2007). This assumption is shown in equation 2.7 below.

$$Output\ Independence\ Assumption: P\ (o_i|q_1 \ldots\ q_i \ldots\ q_T, o_1, \ldots, o_i, \ldots\ o_T) = P(o_i|q_i) \qquad 2.7$$

2.4.3 The Three HMM Problems

In order for the HMM to be useful in building real-world applications of speech recognizers, the following three fundamental problems of Hidden Markov Model must be solved.

1. Evaluation: Evaluating the probability of an observed sequence of symbols
   $O = o_1 o_2 o_3$, given a particular HMM, i.e., $p(O|\lambda)$.

2. Decoding: Finding the most likely state transition path associated with an observed sequence. Let $q = q_1 q_2 \ldots q_t$ be a sequence of states. To find $q = argmax_q\ p(q, O|\lambda)$, or equivalently, $q = argmax_s\ p(q|O, \lambda)$.

3. Training: Adjusting all the parameters λ to maximize the probability of generating an observed sequence, i.e., to Find $\lambda = \text{argmax}\ \lambda\ p(O|\lambda)$.

The first problem is solved by using the forward algorithms. Assume $\alpha_t(j)$ represents the probability of being in state j after seeing the first t observations given the model λ; Thus, the forward algorithm is defined by (Rabiner & Juang, 1993).

1. Initialization

$$\alpha_1(i) = \pi_i b_i(o_1); 1 \leq i \leq N \quad 2.8$$

2. Induction

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(o_{t+1}); 1 \leq t \leq T-1 \text{ and } 1 \leq j \leq N \quad 2.9$$

3. Termination

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \quad 2.10$$

The second problem is solved by using the Viterbi algorithm, also an iterative algorithm to "grow" the best path by sequentially considering each observed symbol.

The optimal state sequence can be found through the procedure below (Rabiner & Juang, 1993).

1. Initialization

$$\delta_1(i) = \pi_i b_i(o_1); 1 \leq i \leq N \quad 2.11$$

$$\psi_1(i) = 0 \quad 2.12$$

2. Recursion

$$\delta_t(j) = \max_{1 \leq i \leq N} [\delta_{t-1}(i)\, a_{ij}] b_j(o_t); 1 \leq j \leq N \text{ and } 2 \leq t \leq T \quad 2.13$$

$$\psi_t(j) = arg \max_{1 \leq i \leq N} [\delta_{t-1}(i)\, a_{ij}]; 1 \leq j \leq N \text{ and } 2 \leq t \leq T \quad 2.14$$

$\psi_t(j)$ Is an array which keeps track of the argument that maximized equation 2.13

3. Termination

$$P^* = \max_{1 \le i \le N} [\delta_T(i)] \quad 2.15$$

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

4. Path backtracking

$$q_t^* = \psi_{t+1}(q_{t+1}^*); t = T-1, T-2, \dots, 1 \quad 2.17$$

The last problem is solved by the Baum-Welch algorithm (an EM algorithm), which uses the forward and backward probabilities to update the parameters iteratively. In order to understand the algorithm, one needs to define a useful probability related to the forward probability, called the backward probability. The backward probability β is the probability of seeing the observations from time t+1 to the end, given state j at time t given the model λ and is defined as (Rabiner & Juang, 1993):

1. Initialization

$$\beta_T(i) = 1; 1 \le i \le N \quad 2.18$$

2. Induction

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)\,;\ 1 \le i \le N \ and \ t = T-1, T-2, \dots, 1 \quad 2.19$$

This time, the forward and backward algorithms can be used to re-estimate the transition (a_{ij}) and observation ($b_j(k)$) probabilities from an observation sequence. The transition probability $\bar{a}_{ij}$ can be estimated as:

$$\bar{a}_{ij} = \frac{\text{expected number of transitions from state } i \text{ to state } j}{\text{expected number of transitions from state } i} \quad 2.20$$

The numerator of equation 2.20 can be computed as the probability $\xi_t(i,j)$ which is defined as the probability of being in state i at time t and state j at time t+1 given the observation sequence.

$$\xi_t(i,j) = P(q_t = i, q_{t+1} = j|O, \lambda) \quad 2.21$$

According to the forward and backward variables, equation 2.21 can be rewritten as:

$$\xi_t(i,j) = \frac{\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)} \quad 2.22$$

Whereas, the denominator (that is the probability of being in state i at time t) is defined as:

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N}\alpha_t(i)\beta_t(i)} \quad 2.23$$

Once all the necessary information is available, the transition probability re-estimate will become:

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1}\xi_t(i,j)}{\sum_{\mathrm{t=1}}^{\mathrm{T-1}}\gamma_t(i)} \quad 2.24$$

The remaining two model parameters will also be re-estimated as follows. The observation probability can be re-estimated as:

$$\bar{b}_j(k) = \frac{expected\ number\ of\ times\ in\ state\ j\ and\ observing\ symbol\ v_k}{expected\ number\ of\ times\ in\ state\ j} \quad 2.25$$

$$\bar{b}_j(k) = \frac{\sum_{\substack{t=1 \\ o_t=v_k}}^{T}\gamma_t(j)}{\sum_{\mathrm{t=1}}^{\mathrm{T}}\gamma_t(j)} \quad 2.26$$

And the initial state probability as:

$$\bar{\pi}_i = expected\ frequency\ in\ state\ i\ at\ time\ (t=1) = \gamma_1(i) \quad 2.27$$

Assuming the initial model being $\lambda = (A, B, \Pi)$ and the re-estimated one $\bar{\lambda} = (\bar{A}, \bar{B}, \bar{\Pi})$, it is proven that $P(O|\bar{\lambda}) \geq P(O|\lambda)$(Rabiner and Juang 1993). This shows that a repeated re-estimation using the new model will improve the probability of the observation sequence, O, being observed given the model.

2.5 Hidden Markov Model Toolkit (HTK)

HTK is the "Hidden Markov Model Toolkit" developed by the Cambridge University Engineering Department (CUED). This toolkit aims at building and manipulating **Hidden Markov Models** (HMMs). It consists of a set of command line tools written in C language to construct various components of a speech recognition system (Young et al., 2009).

2.5.1 Software Architecture

The functionality of HTK is built into the library modules. These modules ensure that every tool interfaces to the outside world in exactly the same way. They also provide a central resource of commonly used functions.

User input/output and interaction with the operating system is controlled by the library module HShell and all memory management is controlled by HMem. Math support is provided by HMath and the signal processing operations needed for speech analysis are in HSigP. Each of the file types required by HTK has a dedicated interface module. HLabel provides the interface for label files, HLM for language model files, HNet for networks and lattices, HDict for dictionaries and HModel for HMM definitions.

All speech input and output at the waveform level is via HWave and at the parameterised level via HParm. As well as providing a consistent interface, HWave and HLabel support multiple file formats allowing data to be imported from other systems. Direct audio input is supported by HAudio and simple interactive graphics is provided by HGraf. HUtil provides a number of utility routines for manipulating HMMs while HTrain and HFB contain support for the various HTK training tools. HAdapt provides support for the various HTK adaptation tools. Finally, HRec contains the main recognition processing functions. Figure 2.4 shows the architecture of the Hidden Markov Model toolkit.

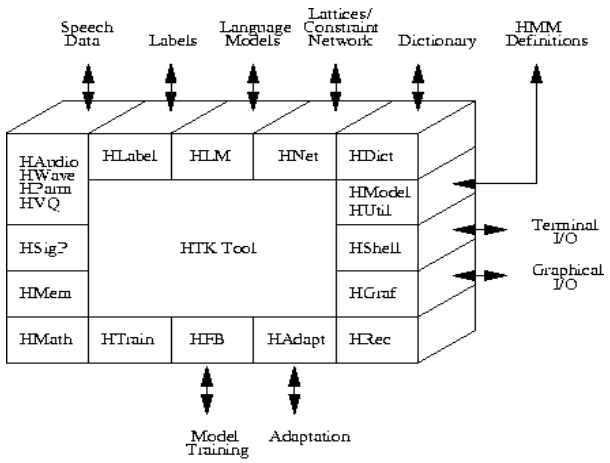

Figure 2. 4 HTK Architecture (Young et al., 2009:15).

There are 4 main phases in the construction of a speech recognizer; data preparation, training, testing and analysis each phase having its own set of tools. Figure 2.5 Shows the tools involved in all the phases.

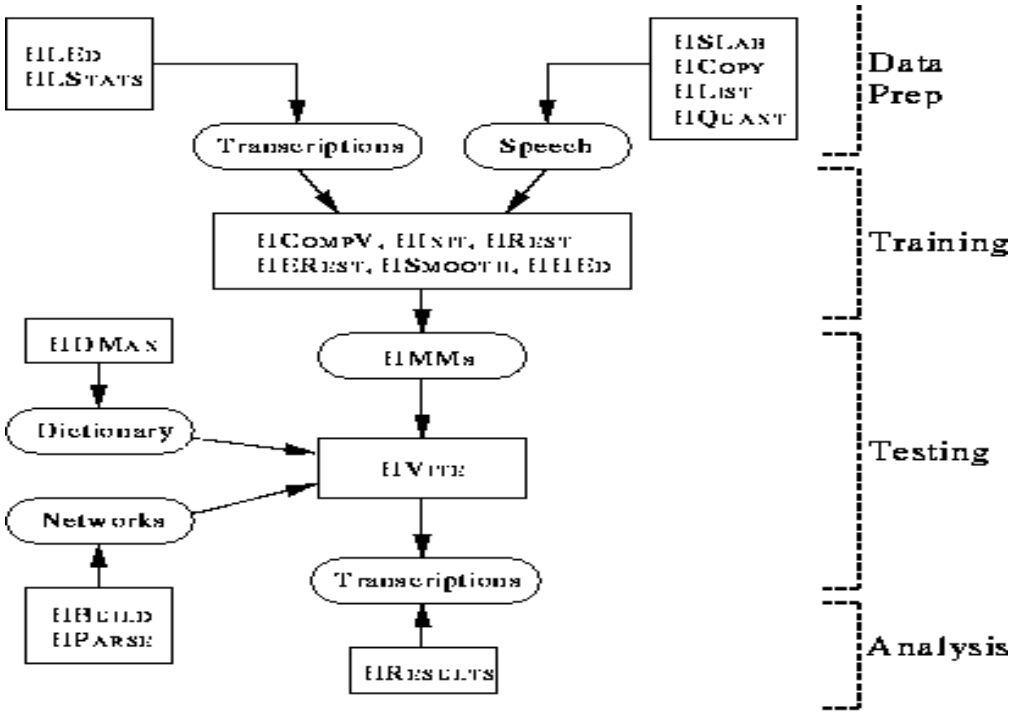

Figure 2. 5 Tools used for developing speech recognition (Young et al., 2008:17).

2.5.2 Data Preparation Tools

Speech data will be obtained from database archives, typically on CD-ROMs. Before it can be used in training, it must be converted into the appropriate parametric form and any associated transcriptions must be converted to have the correct format.

If the speech needs to be recorded, then the tool HSLab can be used both to record the speech and to manually annotate it with any required transcriptions. The tool HCopy is used for the purpose of parameterizing speech data (Young et al., 2009). 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. The tool HLEd is a script-driven label editor which is designed to make the required transformations to label files. HLEd can also output files to a single Master Label File MLF which is usually more convenient for subsequent processing. Finally on data preparation, HLStats can gather and display statistics on label files and where required.

2.5.3 Training Tools

For initialization stage of the training process the tools HInit, HRest and HCompV are used. The tools HInit and HRest provide isolated word style training using the fully labeled bootstrap data. HInit reads in all of the bootstrap training data and cuts out all of the examples of the required phone. On the second and successive cycles, the uniform segmentation is replaced by Viterbi alignment. The initial parameter values computed by HInit are then further re-estimated by HRest. The tool HCompV is used to initialize the parameters of an HMM such that all component means and all covariance's are set equal to the global data mean and covariance which forms the first stage of a flat start training scheme.

Once an initial set of models has been created, the tool HERest performs a single re-estimation of the parameters of a set of HMMs using an embedded training version of the Baum-Welch algorithm. HERest is intended to operate on HMMs with initial parameter

values estimated by HInit/HRest. HERest supports multiple mixture Gaussians, discrete and tied-mixture HMMs, multiple data streams, parameter tying within and between models, and full or diagonal covariance matrices. For performance improvement of specific speakers the tools HEAdapt and HVite are used to adapt HMMs to better model the characteristics of particular speakers using a small amount of training or adaptation data. HEAdapt performs adaptation of a set of HMMs using either maximum likelihood linear regression (MLLR), maximum a-posteriori (MAP) or both (Young et al., 2009).

2.5.4 Testing and Analysis Tools

HTK provides a single recognition/testing tool called HVite which uses the token passing algorithm 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 (Woodland, 2014). HVite then matches a speech file against a network of HMMs and output a transcription for each. When performing N-best recognition a word level lattice containing multiple hypotheses can also be produced.
HBuild is an HTK tool that allows sub-networks to be created and used within higher level networks. HBuild can also be used to generate word loops and it can also read in a backed-off bigram language model and modify the word loop transitions to incorporate the bigram probabilities. The tool HParse is supplied to convert this notation into the equivalent word network. The tool HSGen takes as input a network and then randomly traverses the network outputting word strings. These strings can then be inspected to ensure that they correspond to what is required. HSGen can also compute the empirical perplexity of the task.

To evaluate the performance of an HMM based recognizer the tool HResults is used. HResults tool uses dynamic programming to align two transcriptions and then count substitution, deletion and insertion errors (Young et al., 2009).

2.6 Related Works

As compared to Automatic Speech Recognition, researches done for other Ethiopian languages, it could be said that Amharic ASR is at a better status. There have been numerous researches conducted in the area since 2001. Among the works done, the first one was by Solomon (2001) who developed an isolated Consonant-Vowel syllable recognition system that recognizes a subset of isolated consonant-vowel (CV) syllables. The average recognition accuracy achieved were 87.68% and 72.75% for speaker dependent and independent systems, respectively.

A sub-word based isolated word recognition systems for Amharic using HTK was developed by Kinfe (2002). The sub-word units used in the experiment were phones, tri-phones, and CV-syllables. An average recognition accuracy of 83.07% and 78% was achieved by the speaker dependent phone-based and tri-phone-based systems, respectively. Phone-based and tri-phone-based speaker independent systems had an average recognition accuracy of 72% and 68.4% respectively. Additionally, Zegaye (2003) investigated the possibility of developing large vocabulary, continuous speech and speaker independent Amharic speech recognizer. He built both phone-based and tri-phone based recognizers. The best recognizer, according to Zegaye (2003) was a tri-phone based recognizer which had 76.2% word recognition accuracy. Following which, Solomon (2006) developed a Large Vocabulary Speaker Independent Continuous Speech Recognition System for Amharic using the CV syllables. He has been able to show that syllable models are promising alternatives to phone models by achieving a word recognition accuracy of 90.43% on the 5,000 words evaluation test set.

Adey (2013) Used syllables as longer length acoustic units for the development of an Amharic LVCSRS. This research explored the six Amharic syllable types VC, CV, CVC, VCC and CVCC as acoustic units taking the above concepts into consideration. Adey (2013) also developed a tri-phone based system and compared it to the result obtained from the syllable model which had a higher accuracy of 85.80%. Thus, the researcher concluded that the results obtained showed that the use of all syllable types as acoustic

units does result in a performance increase and that syllables are promising units to use in LVCSR given that enough amounts of training data is used. These are few among the various works conducted in the area.

To the contrary, the only research conducted for Tigrinya ASR was when Hafte (2009) developed a speaker independent continious speech recognizer using 300 sentences recorded by 12 people that are native speakers of the language. He achieved an accuracy of 58.38% and percentage of correctly recognized words 60.32%. A final recommendation made by the researcher was the need for a large vocabulary speech corpus to improve performance as well as develop a robust large vocabulary speech recognizer. Hafte (2009) also attributed the absence of research and slow progress in Tigrinya ASR to the limited amount of speech corpus available for the language.

Review of research works shows that there are international researches in the area of developing a speech recognizer with limited data. The first research reviewed investigated the development of a Persian speech recognizer using limited amount of Persian speech data and borrowing acoustic data from English speech data (Naveen & Shrikanth, 2003). Phoneme mappings from the English phonemes to the Persian phonemes were derived to create seed models for the Persian acoustic models using English speech data. These seed models were further adapted or re-trained using the limited amount of data available in Persian.

In addition to a knowledge (linguistic) based phoneme mapping the researchers introduced a novel data driven phoneme mapping technique based on the Earth Movers Distance (EMD). The speech recognizer built using the sparse Persian speech data resulted in 20.35% error rate whereas the EMD based data driven phoneme mapping technique achieved 19.80% phoneme error rate.

According to the researchers, the techniques used in their research differ from previous techniques in that they used the phoneme mappings (both knowledge driven and data driven) to create seed models which are further adapted/re-trained using the sparse speech

data available in the target Persian language, while most previous techniques proposed the development of acoustic models in the target language using data from all available speech data.

Both the English and Persian phoneme models had 3 states with 16 GMMs per state. The English models were created from the train subset of TIMIT database and Persian model using FARSDAT database, which consists of recordings by 300 Persian speakers from 10 different dialect regions. Transcribed data from 80 speakers which gave approximately 2800 sec of speech were used. Additionally transcribed speech from 18 native Persian speakers (9 females and 7 males), each of whom who read approximately 250 short phrases in the target medical domain were used.

The collected data was used for adaptation/re-training and testing, but not for creation of baseline Persian acoustic models. Since the data available was less, a hold-one-out strategy (or cross validation) was used in the experiments and the models were adapted/re-trained using data from 17 speakers and tested on the remaining one speaker. This was repeated for all 18 speakers and results reported are the average WERs. According to the researchers, the results obtained were very encouraging, illustrating that it is possible to make use of acoustic data even between diverse languages like English and Persian to improve the performance of ASRs in languages constrained by sparse data.

Tanja and Alex (2001) reported the possibility of building LVCSR system for a new target language using speech data from varied source languages, but only limited data from the target languages. The researchers first developed a monolingual recognizer based on a database of 15 languages. According to the researchers, the goal of creating monolingual recognizers in multiple languages was to investigate the differences between languages and highlight resulting challenges for speech recognition in multiple (even less familiar) languages, and as well as explore the adaptation to new target languages. A global unit set which is suitable to cover 12 languages was defined. These language-independent acoustic models allow the data and model sharing of various languages to reduce the complexity and number of parameters of a multilingual LVCSR system.

Acoustic modeling problems were given attention and other resources were assumed to be given in the target language. This assumption was taught to be reasonable because acquiring the training data for acoustic models is usually the most expensive part of a data collection.

The relative effectiveness of language independent acoustic models with a wider context was explored in combination with a polyphone decision tree specialization (PDTS) method. A 19.6% word error rate obtained when adapting language independent acoustic models to Portuguese using only 90 minutes of spoken speech.

An attempt also made by Martin, Lamel, and Adda (2014) to develop the first large vocabulary continuous speech recognition (LVCSR) system for the Luxembourgish language, which is a partially under-resourced language. It suffers from relatively low written production and linguistic knowledge as well as sparse lexical and pronunciation dictionaries. Martin et al. (2014) proposed a LVCSR system which makes use of acoustic models stemming from different major European languages, without making use of Luxembourgish language specific acoustic training data. A phonemic inventory was defined for Luxembourgish and linked to inventories from major neighboring languages (German, French and English), using the IPA symbol set. For each of these languages, acoustic seed models were built using either monolingual German, French, English data or multilingual pooled audio data from the three. Acoustic seed models for a target language n (Luxemburg) is given phone models Pi of languages Li (i = 1; 2; 3:English, French, German) and IPA symbol correspondences between language i and n IPA(i,n).

Three sets of context independent acoustic models were built, one for each well-resourced seed language (i.e. English, French, and German). The models were trained on manually transcribed audio data between 40 and 150 hours from a variety of sources, using language specific phone sets. According to the researchers, the best results were achieved for the acoustic models stemming from the German audio. German ranks best (54.5% WER, 256 Gaussians) before the pooled models (56.6%). English (62.6%) and French (71.5%) produce significantly higher error rates.

The researchers concluded that building acoustic models via IPA associations between the Luxembourgish phonemic inventory and those of other languages for which acoustic models are available gives encouraging results.

Recognition systems developed originally for one language have been successfully ported to several languages, including systems developed by IBM, Dragon, BBN, Cambridge, Philips, MIT and LIMSI. The transformation of English systems to such diverse languages like German, Japanese, French and Mandarin Chinese illustrates that speech technology generalizes across languages and that similar modeling assumptions hold for various languages.

Even though various international works have been done in the area of training a target language using different source languages, local works like designing a Tigrinya speech recognizer, using Amharic speech corpus and Tigrinya data hasn't been done before which makes this work the first attempt.

CHAPTER THREE

The Tigrinya and Amharic Language

3.1 Human Speech Production System

Speech is one of the oldest and most natural means of information exchange between human beings. It has the potential of being an important mode of interaction with computers as well. Speech sounds are produced using, but modifying, the respiratory and longer expiration period. A greater amount of air is expelled, with a gradual decrease in the volume of air and fairly constant pressure. Importantly for the production of sound, the air is often impeded at some point or points on its way out (Brinton, 2000:20).

Speech is produced with the collaboration of the different components of the human speech production system. The main components are: The lungs, trachea (windpipe), larynx, pharyngeal cavity (throat), oral or vocal cavity (mouth), nasal cavity (nose) (Yule, 1996). Figure 3.1 shows the human speech production organs.

Most sounds in human spoken languages are produced when air is expelled from the lung by a downward movement of the ribs and an upward movement of the diaphragm using the intercostal muscles. It travels up the bronchial tubes to the trachea, or"wind pipe", and through the larynx, or "Adam's Apple". The air then moves into the vocal tract, consisting of the oral and nasal cavities (Brinton, 2000:20). The oral cavity, that is, the mouth, is a resonator and a generator of speech sounds via the articulators, which may be active (moving) or passive (stationary).

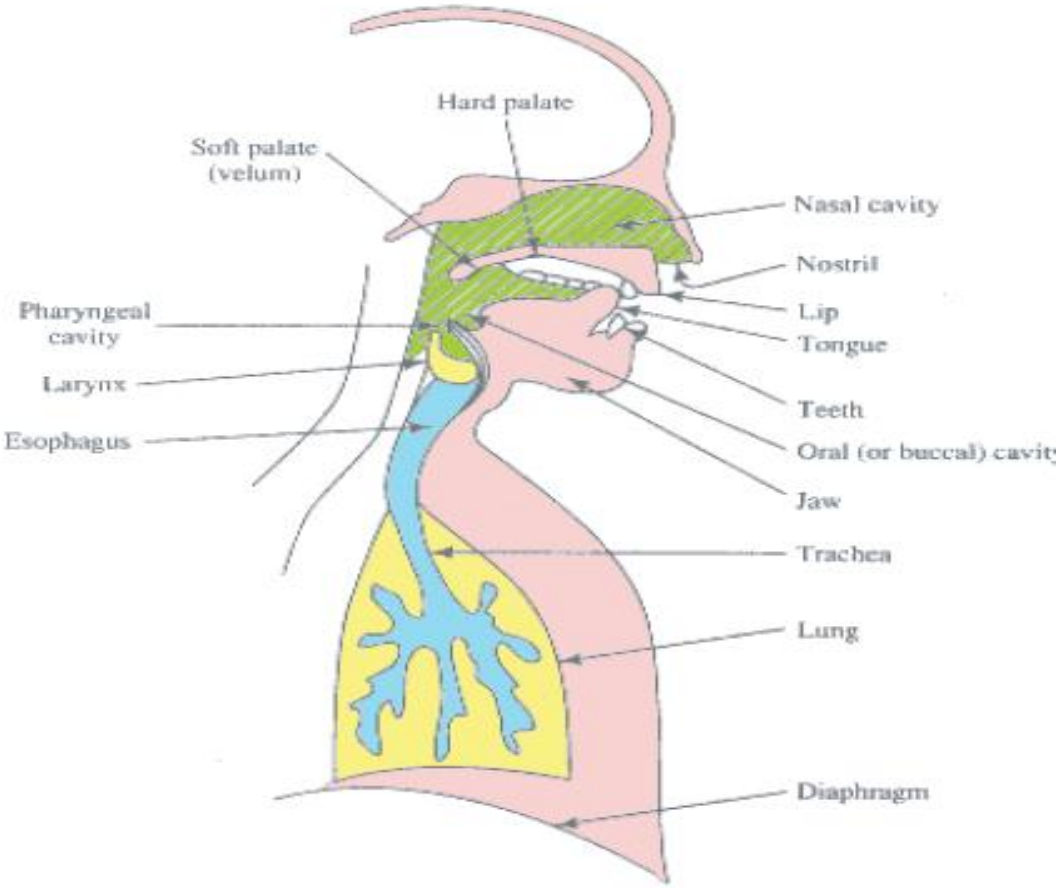

Figure 3. 1 Human Speech Production Systems.[5]

The larynx contains a valve known as the Vocal Cord which are two muscles stretching horizontally across the larynx. The vocal cords are relatively open during normal breathing, but closed during eating. The space between the cords when they are open is known as the glottis. This V-shaped opening between the vocal cords is the most important sound source in the vocal system (Yule, 1996).

The vocal cords may act in several different ways during speech. The most important function is to modulate the air flow by rapidly opening and closing which defines the type of sound unit to be formed. The place of articulation is where the stricture occurs, meaning place of maximum interference and what articulators are involved. Figure 3.2: shows the different position of the articulators.

[5] http://www.careers-india.com 5/24/2014

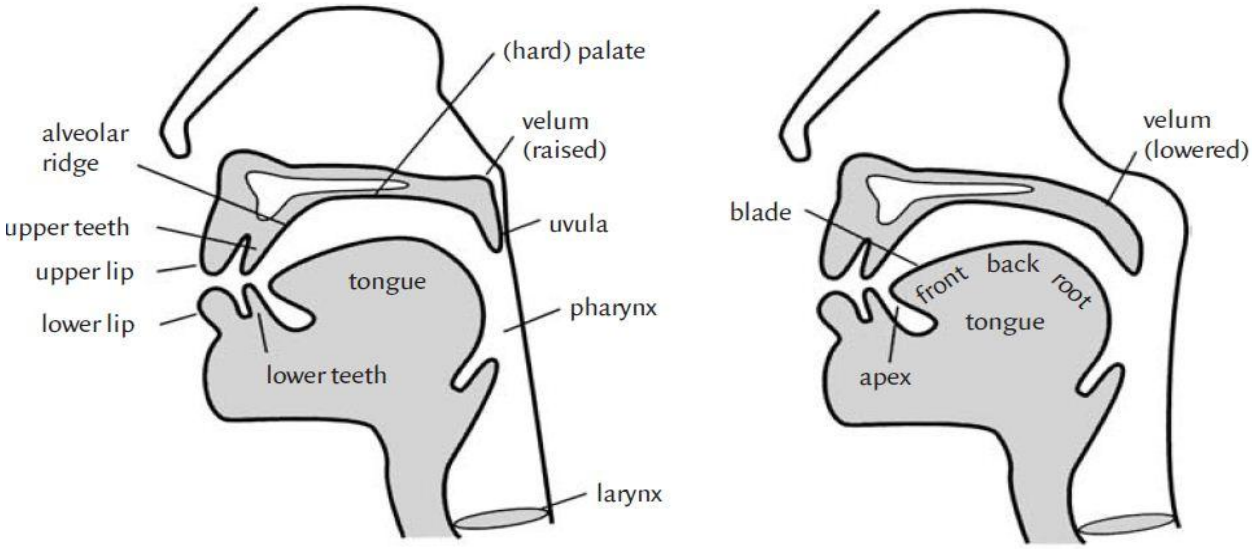

Figure 3. 2 Different Articulator positions (Brinton, 2000:21).

Just as the written form of a language is a sequence of elementary letters, speech is also a sequence of elementary acoustic signals or symbols known as phones that convey the spoken form of a language. The speech output as a result of this production system is a phonetic continuum, a continuous, smoothly flowing set of movements, not a set of discrete and isolated movements (Brinton, 2000:23).

Phones are divided into two main classes called consonants and vowels (Yule, 1996). Both kinds of sounds are formed by the motion of air through the oral, pharyngeal or nasal cavities. Consonants are formed by restricting or blocking the airflow in some way, and they may be voiced or unvoiced. Vowels have less obstruction, are usually voiced, and are generally louder and longer-lasting than consonants (Jurafsky & Martin, 2007).

3.2 Phonetics and Phonology

Phonetics deals with the speech sounds that occur in human speech. It is a study of the description of what speech sounds exist, how the speech sounds are produced and perceived, and what their physical or acoustic properties are (Tesfay, 2002). There are three ways of approaching phonetics. The first one is articulatory phonetics which studies the physiological mechanism of speech production. The second studies the physical properties of sound and is known as acoustic phonetics, while the third called auditory phonetics is the study of the way listeners perceive speech sounds (Yule, 1996:41).

On the other hand, phonology deals with categories and principles that determine the sound patterns in a language. It describes the systems and patterns of speech sound organizations. For instance, it describes the permissible sequences of speech sound organizations in a language (Tesfay, 2002).

3.2.1 The Tigrinya Language

Tigrinya is an Afro-Asiatic language belonging to the Semitic branch and the third widely spoken language in Ethiopia after Amharic and Afan Oromo. It is spoken as a native language by the overwhelming majority of the population in the Tigray region of Ethiopia and in the highland parts of the Eritrea (Kogan, 1997:424).

Tigrinya has an estimated population of 6 million speakers mainly in the Tigray region of Ethiopia and in Central Eritrea (Ager, 2015). The Tigrinya language is also spoken by large immigrant communities around the world, in countries like Sudan, Saudi Arabia, Israel, Germany, Italy, Sweden, the United Kingdom, Canada and the United States.

Knowledge about Tigrinya and Amharic phonetics, their place and manner of articulation is vital in figuring out the relationship between the two languages which helps in the development of the recognizer. The details are described as follows.

Tigrinya Phonetics

As mentioned in section 3.1, nearly all speech sounds are produced by air coming out of the lungs. The air stream originating from the lungs is pushed upwards through the trachea, oral and nasal cavities. On its way, this air stream is modified by the speeh organs.

Most speech sounds differ from each other due to the process of their production, differences in the place and manner production or articulation speech sounds classified into oral and nasal, vowels and consonants, voiced and voiceless, etc. (Tesfay, 2002:20).

In any language, the major differences between consonants and vowels are (Tesfay, 2002:21).

1. The articulation of consonants involves audible obstruction while vowels are produced with no or little obstruction in the vocal tract.
2. Consonants are less sonorous than vowels
3. Consonants take consonant positions i.e. onset and coda position while vowels take nucleus positions in syllables.

Consonants

Tigrinya has a fairly typical set of phonemes for an Ethiopian Semitic language (Ager, 2015). There are a set of ejective consonants and the usual seven-vowel system. The consonants in Tigrinya are divided into bilabial, labiodental, dental, palatal, alveolar, velar, pharyngeal, glottal based on their place of articulation, and into stops, fricatives, affricatives, nasals, liquids , ejectives and glides (semi-vowels) based on manner of articulation Tesfay (2002). Table 3.1 shows place and manner of articulation.

The following definitions and examples of phonemes in place and manner of articulation are from to Tesfay (2002: 21-30).

Place of Articulation

Bilabial- If the lips are brought into proximity or are in contact, bilabial sounds are produced. /b/, /m/, /w/ and /p/ are bilabial sounds.

Labiodental- When the lower lip is raised against the upper incisors, the speech sounds /f/ and /v/ are produced. It is believed that the bilabial /p/ and the labiodental /v/ are borrowed recently from foreign words. E.g. /fərəs/

Dental - when the tip of the tongue approaches or touches the back of the upper incisors, dental sounds are produced. /t/ and /t'/ are examples.

Alveolar - If the tip of the tongue touches or approximates the alveolar ridge, which is found just behind the upper teeth, the alveolar sounds /d/, /s/ and /z/ are produced.

Palatal- when the front part of the tongue is raised towards the hard palate, the place behind the alveolar ridge, the sounds as /ʃ/ and /tʃ/ are produced

Velar – If the back of the tongue is raised towards the velum or soft palate, speech sounds, like /k/ and /g/ are produced.

Pharyngeal – When the root of the tongue is pulled back and approaches the wall of the pharynx, the Tigrinya speech sounds /ħ/ and /ʕ/ are produced.

Glottal – The glottis can be a place of articulation for the Tigrinya sounds /ʔ/ and /h/. The glottal stop /ʔ/ is formed by tightly closing the glottis and suddenly releasing the closure. The laryngeal (glottal) fractive /h/ that produces a rustling sound, nevertheless, is formed by slightly approximating the vocal cords.

Manner of Articulation

Stops – The stream of air moving through the vocal cavity can momentarily be completely blocked off in the mouth or at the larynx and then released. The articulators can be so close to each other that air does not escape between them. A sound involving

this kind of closure is called a stop. Tigrinya has /n/ /m/ /ɲ/ as nasal stops and /b/ /t/ /k/ as oral stops.

Fricatives – These sounds are produced through a narrow passage, causing friction, when the contact between the active and passive articulators is not close enough to block a continious flow of air in the oral cavity also in the pharynx /ħ/, /ʕ/ and larynx for /*h*/. Such consonants of Tigrinya are /f/,/s/,/z/,/ʃ/, /ʒ/ etc.

Affricates – When a sound such as /dʒ/ are produced, a slow release of closure is shown. Such consonants are called affricates. They are produced first like a stop and end like a fricative. Tigrinya has /tʃʾ/, /dʒ/, /tʃ/ and /s'/ as affricates.

Nasals – These sounds are produced when air escapes through the nose. The nasals in Tigrinya are /m/,/n/ and /ɲ/.

Trill – Among the speech sounds of Tigrinya are /l/ and /r/ which are called liquids. There is some obstruction in the production of these sounds but it is not strong enough to produce friction.

Glides (semi-vowels) – Glides are transition sounds that my be called semivowels. When producing these sounds, no obstruction of the air stream in the oral cavity is observed however there is approximation of articulatory organs. On the other hand, they differ from vowels in that they have a consonantal position in a syllable. Tigrignna has /w/, /y/, /ʔ/ and /h/ as glides.

Ejectives (also known as explosives) – There are different ejectives (also known as explosives or glottalized consonants) in Tigrinya. Ejectives are egressive, not ingressive, because they are produced when the air in the mouth is pushed out. In producing ejectives, the glottis is tightly closed and the larynx is raised. Such consonants of Tigrinya are /p'/, /t'/, /s'/, /tʃ'/,/k'/,/ k'^{w} /,/x'/ and /x$^{'w}$/.

Place of Articulations										
Manner of Articulation			Bilabials	Labiodental	dentals or alveolars	palatals	velars	Labiovelars	pharyngeal	Glottal
	Stop	Voiceless	p		t		k	k^w		ʔ
		Voiced	b		d		g	g^w		
		Ejectives	p`		t`		k’	$k’^w$		
	Fricatives	Voiceless		f	s	ʃ	x	x^w	ħ	h
		Voiced		v	z	ʒ			ʕ	
		Ejectives					x’	$x’^w$		
	Affricates	Voiceless				tʃ				
		Voiced				dʒ				
		Ejectives			s`	tʃ’				
	Nasals	Voiced	m		n	ɲ				
	Trill	Voiced			l r					
	Glides	Voiced	w			y`	w^{**}			

Table 3. 1 Tigrinya Consonants (Tesfaye, 2002:25).

/w/ is indicated in two places in the chart shown above because it is articulated with both narrowing the lip aperture and with raising the tongue towards the soft palate which makes it bilabial and velar respectively.

Fricative /x/, /x^w/, /x’/, and /$x’^w$/ are allophones of stops /k/, /k^w/, /k’/, and / $k’^w$ / phonemes occur after a vowel, respectively. As it is shown in the table difference between them is that the former ones are fricatives while the latter are stops. These stops become fricatives when their un-geminated form is preceded by a vowel (Tesfay, 2002:26).

Vowels

Tigrinya has seven vowel phonemes that distinguish word meanings. In the production vowels, the articulators do not come close to each other and the air flows freely through the vocal tract. Vowel sounds, unlike consonants can stand alone (Tesfay, 2002:30).

These vowel sounds are specified in terms of the configuration of the lips and the position of the tounge. In other words, vowels can be described in terms of the degree of lip rounding, front-back position of the tounge and the relative position of the highest points of the tongue body. Table 3.2 shows the seven vowels of Tigrinya.

	Front	Central	Back
High	i	ɨ	u
Mid	e	ə	o
Low		a	

Table 3. 2 Tigrinya Vowels.

Front vowels= /i/, /e/

Back vowels= /u/, /o/

High vowels=/i/,/ɨ/,/u/

Low vowel= /a/

Central vowels = / ɨ /,/ə/,/a/

Mid vowels = /e/, /ə/, /o/

3.2.2 The Amharic Language

Amharic is the working language of the government of Ethiopia. Since the 13th century, it has been the language of court and has been dominant in Highland Ethiopia. It is used in government office, court system, and for official documents. Amharic has close to 22 million first-language speakers and 4 million second-language speakers worldwide, of which slightly over 21.6 million live in Ethiopia (Phillips, 2013).

In 1994, The Education and Training Policy of the Government of Ethiopia gave all ethnic groups the right to develop and use their language as medium of instruction in primary schools. Secondary schools and universities use English, as the primary medium of instruction (Sharma, 2013).

Amharic Phonetics

Consonants

According to Baye (2000E.C.) Amharic has a total of 30 consonants. However Leslau (2000), states that amharic has a total of 31 consonants with an additional one. Just like consonants in other syllabic languages the consonants in Amharic are further devided based on their manner and place of articulation. Table 3.3 shows the consonants found in Amharic.

Place of Articulations								
Manner of Articulation			Bilabial	Dental - Alveolar	Palatal	Velar	Labiovelar	Glottal
	Stop	Voiceless	p	t		k	k^w	ʔ
		Voiced	b	d		g	g^w	
		Ejectives	p`	t`		k’	$k^{w'}$	
	Fricatives	Voiceless	f	s	ʃ			h
		Voiced		z	ʒ			
		Ejectives		s`				
	Affricates	Voiceless			tʃ			
		Voiced			dʒ			
		Ejectives			tʃ’			
	Nasals	Voiced	m	n	ɲ			
	Liquids	Voiced		l r				
	Glides	Voiced	w		y			

Table 3. 3 Amharic Consonants (Baye, 2008:7).

The consonants s,z and s are also called “sibilants”. v and p appear only in modern loanwords like visa and police. However, the ejective s or glottalized sounds (t’ k’ p’s’ tʃ’) are peculiar to Amharic.

Vowels

Similar to the vowels found in Tigrinya, there are seven vowels in Amharic.

	Front	Central	Back
High	i	ɨ	u
Mid	e	ə	o
Low		a	

Table 3.4 Amharic Vowels.

3.3 Writing System

To express thoughts through writing, symbols that represent sounds or phonemes are needed. This art of expressing thoughts through symbols is called writing and the nature of writing as writing system. Writing systems used to represent the world's languages fall into two major categories: ideographic and phonologic. In the ideographic system is like the Chinese system where, characters represent the meaning of a word. Phonological system on the other hand is divided into syllable system, like in Japanese, Kana, and alphabetic used in most Indo-European languages, such as Latin, English and German (Yule, 1996).

In the syllabic system**,** a symbol represents a combination of a vowel and a consonant. In this system, the numbers of symbols needed for a given language is determined by the number of basic sounds used (Daniel, 1997:16).

Both Tigrinya and Amharic use the syllabic system. The graphemes used in Tigrinya and Amharic are known as "Fidel". Each unit represents a consonant + Vowel imprint the basic unit is the consonant combined with each of the seven vowels to form the following syllable (Hafte, 2009).

Transcription:	/h-ə/	/h-u/	/h-i/	/h-a/	/h-e/	/h-ɨ/	/h-o/
Orders :	1st	2nd	3rd	4th	5th	6th	7th
Grapheme Symbol:	ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ

The orthography of Tigrinya does not mark gemination, so the pair of words *k'ärräbä* 'he approached', *k'äräbä* 'he was near' are both written as "ቀረበ". Since such minimal pairs are very rare, the problem caused for readers is little (Hafte, 2009).

In addition, there are twenty numerals and eight punctuation marks in the writing system. However, research in speech recognition deals with distinct sounds only.

Phonological scripts are easier than ideographic scripts in speech recognition framework. In many cases, rule-based letter-to-sound mapping tools can be used to generate the pronunciation dictionary needed to guide recognition this is usually not possible in ideographic systems.

3.4 Comparison of Tigrinya and Amharic

According to Tesfaye (2002) Tigrinya has a total of 39 phonemes out of which 32 are consonants and 7 of them vowels. Among the total 39 phonemes, 2 are unique to the language and not found in the Amharic language. The remaining 37 phonemes are similar to that of the Amharic.

Table 3.5 shows a combined phonemic chart of Amharic and Tigrinya. If phoneme is present in both Tigrinya and Amharic, it is simply entered in the chart in the appropriate position. In cases where the status of a phoneme is nonexistent or marginal, it is shown in parentheses. (Shimelis, 2014). All of the phonemes in Tigrinya and Amharic are the same with the exception of the pharyngeal fricatives ħ and ʕ found only in Tigrinya.

	Bilabial	Labio-dental	Alveolar	Post-alveolar	Palatal	Velar	Pharyngeal	Glottal
Plosive	p b p’		t d t’		tʃ dʒ tʃ’	k k^w g g^w k’$k’^w$		ʔ
Nasal	m		n		ɲ (t)			
Trill			r					
Fricative		f	s z s’	ʃ ʒ(t)			ħ (a) ʕ (a)	h
Approximant	w^*				y			
Lateral approximant			l					

Table 3. 5 Tigrinya and Amharic Phonemes (Shimelis, 2014:4).

*/w/ results from two simultaneous articulations, bilabial and velar; hence, labio-velar approximant. Here it is called bilabial. The notation involves the letters *a* and *t* (in that order), indicating respectively “Amharic and Tigrinya”.[6] (Shimelis, 2014).

[6] *a, t :* lower case means the phoneme is marginal in the language (a phoneme that is normal in a language is not marked) e. g ʒ(t) is normal in Amharic, marginal in Tigrinya

a, *t*: underlined lower case means the phoneme is nonexistent in the language

Additionally, the convention is adopted that, when voiced, voiceless and ejective phoneme counterparts exist, the voiceless sound is written on the left, the voiced sound on the right and the ejective below both of them, in a triangle configuration (Shimelis, 2014:4).

CHAPTER FOUR

Experimentation

4.1 Introduction

The LVCSR for Tigrinya which is HMM based was developed using the Hidden Markov Model toolkit installed on a Linux platform. Language model development was done using the SRILM toolkit. Various other preprocessing programs and scripts were also used for the purpose of manipulating the raw text data such as python scripts for preparing the pronunciation dictionary and bringing the transcription data into one format. The discussions of the experiments are presented in accordance to the steps that have been followed while building such speech recognizers.

4.2 Architecture of the Recognizer

Figure 4.1 illustrates a training corpus containing the speech data along with its transcription. The train speech data is first converted into the parameterized format. The transcription file is then converted into the word level transcription. Using the train lexical model/pronunciation dictionary and the word level transcription, phone level and tri-phone level transcription files are created.[7] The Acoustic model is trained using the tri-phone level transcriptions and the parameterized speech data. The language model is developed using a separate Tigrinya text data collected for that purpose. Finally the test speech data is first converted into the parameterized speech format and given as an input to the recognizer which comprises the three models; the language model, the acoustic model and the decoding lexical model.

[7] Word level transcription means reducing the sentence into one word per line whereas phone level transcription means reducing the words into its phonetic equivalent, one phone per line.

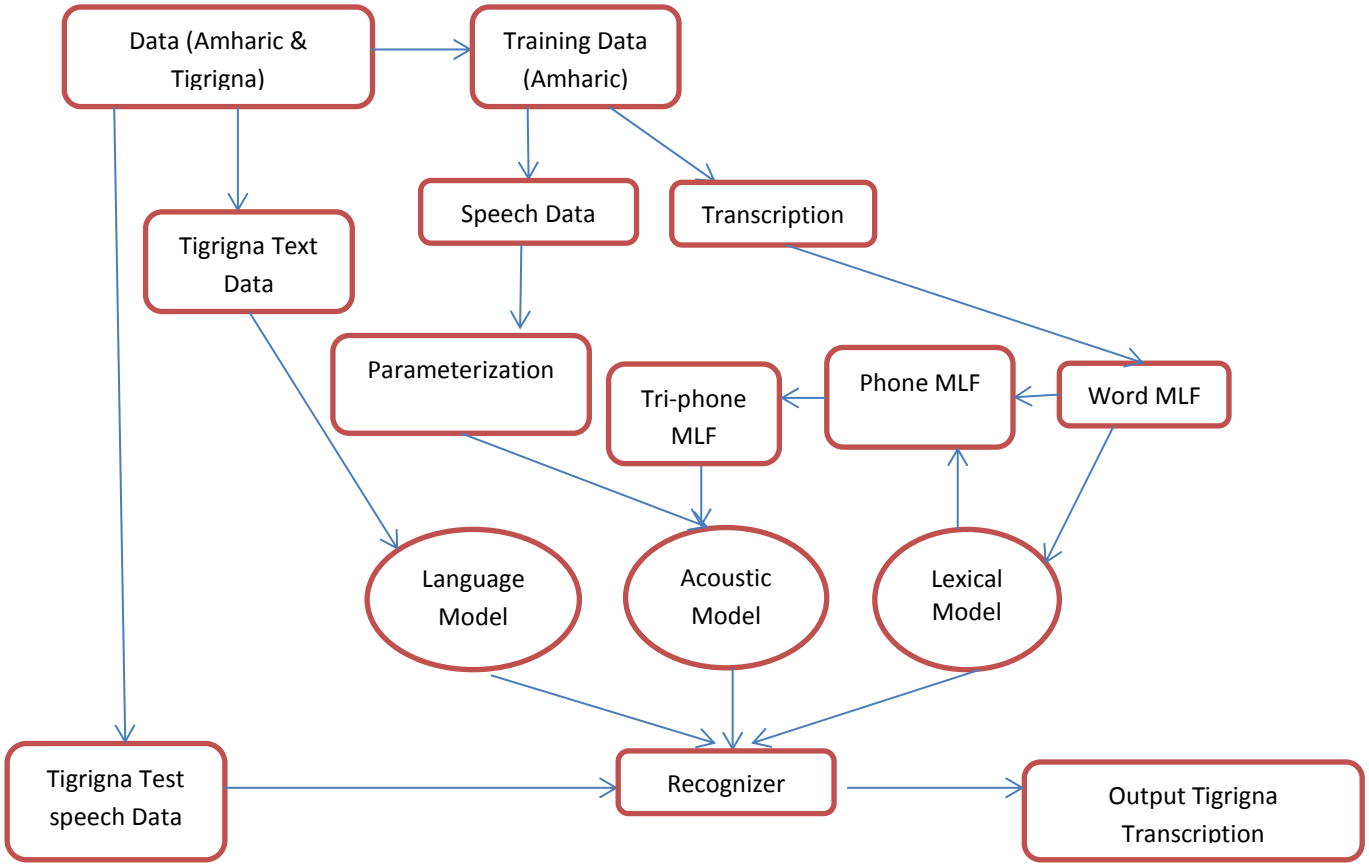

Figure 4. 1 The architecture of Tigrinya speech recognizer designed in this study

4.3 Data Set Used for Training and Testing

Developing the Tigrinya recognizer using a different source language required Amharic speech data for the purpose of training the acoustic model and Tigrinya speech data for adapting and testing the model. Developing a recognizer requires a language model in addition to the acoustic model to make it complete. Developing a language model requires text data from the target language the recognizer is going to output. Thus, there was a need to collect large amount of Tigrinya text data to train the language model.

4.3.1 Amharic Speech Corpus

As described in section 1.5.2 this research made use of a previously developed Amharic speech corpus prepared by Solomon et al. (2005) for LVCSR. It contains 20 hours of training speech collected from 100 speakers who read a total of 10,850 sentences. However, one modification was made to the data. The format used to represent the transcription file in the Amharic corpus has been changed to the format used in the transcription files for the Tigrinya data.

The HTK toolkit doesn't support Unicode format thus there was a need to transliterate all the speech and language model text, although the text in the Amharic speech corpus is already transliterated.

4.3.2 Tigrinya Speech Corpus

The Tigrinya speech corpus prepared was 2 hours of read speech data containing 600 sentences together with the train and test data. The text from which the sentences were read or the prompt sentence was collected from various news websites found via the Internet. A total of 1000 sentences were extracted from these news websites out of which 600 were chosen randomly to be further processed and read. The sentence length ranged from a minimum of 4 words to a maximum of 16 words.

The chosen text for the 600 sentences was cleaned by correcting grammatical and spelling errors, removing repeated words and transcribing numbers into texts. Once the data was processed and ready to be recorded, 12 people comprising of age groups from 24 – 45 were selected to read the text. All of the chosen readers have the educational background of first degree and above.

Having tackled the preparation stage which needed careful preparation of the prompt data to be read, the actual recording process was performed in a radio studio located at Addis Ababa University Department of Communication and Journalism. The radio studio contained state of the art recoding equipment's which facilitated the whole recording

process and contributed to the good quality of the recorded data. Factors like environmental noise disturbances that affect the quality of a speech corpus were minimized or reduced to a near zero as the recording was done in a professional recording studio. Noise canceling microphones also played a role in keeping the quality of the recorded speech up to standard. . Recording the speech was done using praat software.[8] Table 4.1 presents summary of training and test data set used for the experiment.

Corpus	Number of Speakers	Male	Female	Number of Sentences	Number of Tokens
Adapt/Train	6	5	1	300	3,324
Test	6	5	1	300	3,354

Table 4. 1 Description of Tigrinya Speech Corpus.

Additionally, as mentioned in section 1.5.2, 300 sentences prepared by Hafte (2009) were also used for adaptation purpose in addition to the recorded speech data.

4.3.3 Tigrinya Language Model Text

To train the Language Model, Tigrinya text containing 50,000 sentences or 1,341,578 tokens has been collected from different sources, most of them are news reports. As the collected text was originally written in the Ethiopian alphabet "fidel", there was a need to transliterate it into Latin format. The transliteration of the Tigrinya text was also done here because the HTK toolkit doesn't support Unicode format.

[8] Praat is a freeware program used for the analysis and reconstruction of acoustic speech signal.

4.4 Preprocessing

4.4.1 Dictionary

Training and testing dictionary are needed to train and test the recognizer. Thus the first step of the preprocessing stage is to develop training and test dictionary that would later be used for the development process. To develop the train dictionary, a world list has been created from the 600 Tigrinya training sentences and the 10,850 Amharic training sentences. The word list contains a combination of Amharic and Tigrinya list of unique words that are found in the training sentences sorted in an alphabetical order. A perl script (prompt2wordlist) was used to create the wordlist. Once the wordlist was created, the following HTK command HDMan was run to create the actual dictionary.

However, depending on the type of experimentation done the training dictionary vary. In experimentations done with only Amharic data, the training dictionary only incorporates the unique Amharic words whereas when a combination of the two languages is used to train the models, a dictionary containing words from both languages has been used.

```
HDMan -m -w Wordlist/wlist -n Phones/monophones1 –l log/dlog
Dictionary/traindictionary Dictionary/sourcedictionary
```

HDMan takes as an input the wordlist that was created and a source dictionary which contains the list of words together with their phonetic transcription. The source dictionary was prepared using a python script which takes the wordlist as an input and reduces it to a list of phonetic transcriptions because each grapheme represents an assimilation of a consonant and a vowel. The dictionary that was created is attached in appendix B.

Following the same set of steps and the wordlist from the test transcription data, a test dictionary was also created for evaluating the performance of the speech recognizer.

4.4.2 Transcription File

To train the set of phone HMM models two types of phone level transcription files are required. The first phone level transcription file used to develop the initial phone models was generated without a short pause (sp) in between words. After certain iterations of training has been done, short pause model was introduced to handle any silences the speaker might have in between words. Thus, another phone level transcription file which contains a sp in between each word was also generated.

For the phone level transcription file to be generated, a word level MLF file is first required. Using the orthographic transcription for all the training sentences (10,850 Amharic and 600 Tigrinya) and the perl script file (prompts2mlf) the following word level MLF (words.mlf) file was generated.

```
#!MLF!#
"*/s502.lab"
sil
EnEtayE
rEiihhazEbElE
HEto
ane
kEHatomE
Enategebeanii
qediimomEnii
sil
.
```

Each word in the word level MLF file should be written in a single line and each utterance should be terminated by a single period. The first line of the file just identifies the file as a Master Label File (MLF), which is a single file containing a complete set of transcriptions for all the training sentences (Amharic and Tigrinya). Using the word level MLF file generated above, the following phone level MLF (transcription) file was created with the label editor HLed.

HLEd -l '*' -d Dictionary/traindictionary.txt -i MLF/phones0.mlf Config/mkphones0.led MLF/words.mlf

The HLEd takes as an input the train dictionary (traindicitonary.txt) and word level MLF (words.mlf) files generated above to create the phone level transcription file phones0.mlf. For each word that HTK finds in the word level MLF it looks for the same word in the dictionary to replace it with its phonetic transcription. Depending up on the type of edit script used phone level mlf with and without sp can be generated. The phone mlf files phone0.mlf and phone1.mlf are shown below:

Phones0.mlf

```
#!MLF!#
"*/s502.lab"
sil
E
n
E
t
a
y
E
r
E
ii
hh
a
z
E
b
E
l
E
sil
.
```

Phones1.mlf

```
#!MLF!#
"*/s502.lab"
sil
E
n
E
t
a
y
E
sp
r
E
ii
hh
a
z
E
b
E
l
E
sil
.
```

4.4.3 Coding Speech Files

The final stage of data preparation is to parameterize the speech files which are recorded in the wave format into a sequence of feature vectors. The Mel Frequency Cepstral Coefficients (MFCCs) was used to do this.

The HTK tool used to perform the conversion was the HCopy tool which automatically converts its inputs into MFCC vectors, as shown below.

```
HCopy -T 1 -Configs/config.txt -Scripts/train_change_script.txt
```

The file config.txt specifies the type of parameters to be used when changing the speech files into MFCC format. It specifies that the target parameters are MFCC using Co as the energy component, the frame period is 10msec that the output should be saved in compressed format and a crc checksum should be used. Additionally, it specifies that FFT should use a Hamming window and that the signal should have first order preemphasis applied using a coefficient of 0.97. The filterbank should have 26 channels and 12 MFCC coefficients should be output. The config file used is attached in appendix C.

Next to the config file, the train_change_script.txt file contains the path and file names of all the speech files and the name of the MFCC file to be used after conversion. The script file used is shown below:

```
/Data/Train/amharic/tr01001.wav        /Data/Train/Amharic/tr01001.mfcc
/Data/Train/tigrinya/s1.wav            /Data/Train/Tigrinya/s1.mfcc
```

Thus, running the HCopy tool in the above way generated parameterized vectors (MFCC) for all the speech files which were in the wave format and that would later be used for training and testing the recognizer.

4.5 Language Model Training

The Language Models used was a bigram and a trigram language model. It was developed using the corpus developed in section 4.3.3 by using the SRILM toolkit. Soothing techniques like the modified Kneser-Ney were used. The following SRILM command was run to develop the Bigram language model.

> Ngram-count -text languagemodeldata.txt -order 2 -lm /LModel/lm2.txt -kndiscount2 -vocab Wordlist/testwordlist.txt

The HTK HBuild tool was used to translate the language model developed using SRILM into a standard lattice format. It is a format used to build and represent the word network used in the decoding phase by HVite. The following Hbuild command was used:

> Hbuild –n langkn.txt testwordlist.txt wdnetwork.txt

The following SRILM command was run to develop the trigram language model.

> Ngram-count -text Thesis/languagemodeldata.txt -order 3 -lm Thesis/LModel/lm3.txt -kndiscount2 -vocab Thesis/wordlist/testwordlist.txt

Having developed the trigram LM there was no need converting it to a word network as the decoding tool HDecode doesn't use the language model directly. Whereas in the case of Bigram LM there was a need to convert it into word network because the decoding tool Hvite only accepts SLF format.

4.6 HMM Model Training

The data preprocessing stage presented in the previous section outputs all the resources needed to start the training process. Transcription files, pronunciation dictionary, train and test script files and parameterized speech vectors have been prepared. This section will illustrate the processes undergone in the development of HMM based acoustic models and also the techniques followed in refining and improving performance of the models. It will also comprise the steps taken to train the language model. Now the training proceeds to create HMM based acoustic and language model.

4.6.1 Prototype Model

Before starting to train the set of HMM models, the first step in HMM training is to define a prototype model whose definition would later be used as a basis for all the models to be developed. This prototype model allows the definition of the model topology, parameter kind, number of states and allowable transitions between states.

The mean and variance parameters defined in the prototype model are ignored with the exception of the transition probabilities, as the purpose of the prototype definition is to only specify the overall characteristics and topology of the HMM. The actual parameters are estimated during the training process.

When choosing an HMM topology, different factors have to be considered. The unit of recognition and the amount of the training speech data available have to be well thought out. This is because as the size of the recognition unit increases and the size of the model in terms of the number of parameters to be re-estimated grow, the model requires more training data (Young et al., 2009).

For a phone-based system, a good topology to use is a 3-state Bakis model with no skips (Young et al. 2009). Thus, the topology chosen for this research is also a 3 state Bakis model topology with no skips. The prototype model used is attached in appendix D.

4.6.2 Initialization

Once the prototype model is constructed and the model topology decided, training starts with initialization. Initialization is the process by which an initial estimate values are obtained for the transition and observation probabilities of the prototype HMM model.

The initialization of a set of phone HMMs before embedded re-estimation is performed using a bootstrap method or flat start method. The bootstrap method requires a set of labeled transcription for each of the speech data available to initialize the model. The flat start method which is a much simpler initialization procedure uses HCompV to assign the global speech mean and variance to every Gaussian distribution in every phone HMM. This procedure implies that during the first cycle of embedded re-estimation, each training utterance will be uniformly segmented. The hope then is that enough of the phone models align with actual realizations of that phone so that on the second and subsequent iterations, the models align as intended.

Labeling is difficult to do as the amount of training data increases. And since labeling the data to be used in these experiments would have been difficult, the method of initialization chosen is the flat start method which uses the HCompV command.

The HCompV tool scans the set of all parameterized speech data files, computed the global speech mean and variance and set all of the Gaussians in the prototype phone models to have the same mean and variance. The command below was executed to do the initialization:

```
HCompV -C Configs/config.txt -f 0.01 -m -S Scripts/trainscript.txt -M HMM/hmm0
HMM/hmm/proto
```

The command created a new initialized version of the prototype model ‘proto’ and stores it in the directory hmm0 in which the zero means and unit variances were replaced by the global speech means and variances.

After initializing the new prototype model, a copy of it is made manually for each of the 40 phonemes for which the models are going to be trained including sil. The copies made are stored in hmm0 directory and stored as a Master Macro File (MMF) which contains all of the copies of the prototype model in one file. The files are stored in an MMF format because it avoids having a large number of individual HMM definition files.
The 39 unique phones were generated from all of the dataset including both Amharic and Tigrinya.

4.6.3 Embedded Re-estimation

Once the initialized phone model set was created for all the 40 phones available, re-estimation needs to be applied. The main HMM training procedures for building sub word systems is done by embedded training (re-estimation).This process simultaneously updates all of the HMMs in a system using all of the training data. Re-estimation process involves updating the initialized transition probabilities and observation likelihood parameters for each of the 40 models in the set. This process in HTK is done by the tool HERest which uses the Baum-Welch re-estimation algorithm.

HERest first loads in the complete set of initialized phone HMM definitions stored in the MMF hmm as well as the training file into the memory turn by turn. It then uses the associated phone level transcription of each training data to construct a composite HMM which spans the whole utterance. The Forward-Backward algorithm is then applied and the sums needed to form the weighted averages are accumulated. Once all the training files have been processed, the new parameter estimates are formed from the weighted sums and the updated HMM set hmm is saved in the directory hmm1.

Thus, the following HERest command was executed to do the embedded re-estimation on the 40 initialized HMM phone models.

HERest -C Configs/config.txt -I MLF/phones0.mlf -t 250.0 150.0 1000.0 -S Scripts/trainscript.txt -H HMM/hmm0/macros -H HMM/hmm0/hmm -M HMM/hmm1 Phones/tphones0.txt[9]

The above command loads all the phone models stored in hmm0 directory which are listed in the phone model list tphones0.txt. The models loaded are then re-estimated using the parameterized speech data listed in trainscript.txt and the phone level transcriptions stored in phones0.mlf. The new re-estimated model set is then stored in the directory hmm1. The -t option sets the pruning thresholds to be used during training.

Each time HERest is run it performs a single re-estimation. Thus following this, the execution of HERest was repeated twice more until the output directory of the new HMM set was stored in hmm3.

4.7 Fixing the Silence Model

In the previous section, a 3 state left-to-right HMM for each of the 39 phonemes and also an HMM for the silence model sil has been generated. Following which extra transitions from states 2 to 4 and from states 4 to 2 were added in the silence model. Adding this extra transition makes the model more robust by absorbing the various impulsive noises in the training data. The backward skip (4 to 2) in this extra transition allows this to happen without committing the model to transit to the following word. Additionally, a 1 state short pause sp model was created. This sp model also called the tee-model has a direct transition from entry to exit state and has its emitting state tied to the center state of the silence model.

At first a text editor was used to copy the HMM set and the center state of the sil model to make a new sp model and store the resulting model in the MMF hmm, which includes the new sp model. This new file ‘hmm’ which contained the sil, sp and 39 phone models was stored in the new directory, hmm4.

[9] tphones0.txt file contains the 40 phonemes for which the models have been constructed as well as the sil model.

The following command was then executed to add the extra transitions required in the sil model and tie the sp model state to the center state of the sil model:

```
HHEd -H HMM/hmm4/macros -H HMM/hmm4/hmm -M HMM/hmm5 Configs/sil.hed
Phones/tphones.txt
```

The file sil.hed used in the HHEd command above contains the following commands which are responsible for adding the transitions and tying the center states.

```
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 models were then re-estimated using the phone transcriptions with sp models (phones1.mlf) between words, resulting in the better re-estimated phone model set stored in hmm.

4.8 Creating Tied-State Triphones

Having developed the set of 41 monophone HMMs, the final stage of model building was to create context-dependent triphone HMMs. This was done in two steps. First, the monophone transcriptions were converted to triphone transcriptions and a set of triphone models were created by copying the monophones and re-estimating. Second, similar acoustic states of these triphones were tied to ensure that all state distributions can be robustly estimated without shortage of training data.

4.8.1 Making Triphones from Monophones

Context-dependent triphones are made by simply cloning monophones and then re-estimating using triphone transcriptions. The list of unique triphones and triphone transcriptions are created by running the following HLEd command:

```
HLEd -n Triphones/triphones1 -l '*' -i MLF/wintri.mlf Configs/mktri.led MLF/phones1.mlf
```

The HLEd command converts the monophone transcriptions in phones1.mlf to an equivalent set of triphone transcriptions in wintri.mlf at the same time, a list of triphones is written to the file triphones1.

The edit script mktri.led contains the commands WB sp , WB sil and TC which defines sp and sil as the word boundary commands. The cloning of the 41 HMM monophone models to tri-phone models can be done efficiently using the HMM editor HHEd:

```
HHEd -H HMM/hmm9/macros -H HMM/hmm9/hmm -M HMM/hmm10 Configs/mktri.hed Phones/tphones.txt
```

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 Triphones/triphones1
TI T_ah {(*-ah+*,ah+*,*-ah).transP}
TI T_ax {(*-ax+*,ax+*,*-ax).transP}
```

The file mktri.hed was generated using the *Perl* script (maketrihed) which is included in the HTKTutorial directory. Once the context-dependent models have been cloned, the new triphone set can be re-estimated using HERest. This is done as the previously estimations were done except that the monophone model list is replaced by a triphone list and the triphone transcriptions are used in place of the monophone transcriptions. Re-

estimation was done again, so that the resultant model sets will ultimately be saved in hmm12.

```
HERest -C Configs/config.txt -I MLF/wintri.mlf -t 250.0 150.0 1000.0 -s Stats/stats -S Scripts/trainscript.txt -H HMM/hmm11/macros -H HMM/hmm11/hmm -M HMM/hmm12 Triphones/triphones1
```

Word internal and cross word tri-phones were generated for the bigram and trigram language models respectively.

4.8.2 Making Tied-State Triphones

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 triphone. The decision tree attempts to find those contexts which make the largest difference to the acoustics and which should therefore distinguish clusters. The following HHEd command was run to perform Decision tree state tying:

```
HHEd -B -H HMM/hmm12/macros -H HMM/hmm12/hmm -M HMM/hmm13 Configs/tree.hed Triphones/triphones1 > Thesis/log
```

The edit script tree.hed adapted from Hafte (2009), contains the instructions regarding which contexts to examine for possible clustering which is a long and complex process. Once all state-tying has been completed and new models synthesized, some models may share exactly the same 3 states and transition matrices and are thus 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 tied.list. The edit script adapted from Hafte (2009) is attached in appendix E.

The log file will include summary statistics which give the total number of physical states remaining and the number of models after compacting. Finally, and for the last time, the models were re-estimated twice using HERest. The trained models are then contained in the file HMM/hmm15/hmm.

```
HERest -C Configs/config.txt -I MLF/wintri.mlf -t 250.0 150.0 1000.0 -s Stats/stats -S Scripts/trainscript.txt –H HMM/hmm14/macros -H HMM/hmm14/hmm -M HMM/hmm15 Triphones/triphones1.txt
```

4.9 HMM Model Refinement and Optimization Using Mixture Splitting

One of the common methods of HMM model refinement and optimization to achieve better performance is mixture component splitting. Multiple mixture systems are said to improve recognition results considerably because they help avoid the problem resulting from the usage of the same type of distribution for different models and different states. If an HMM state is made to contain multiple mixture components, then the training vectors would be associated with the highest likelihood mixture component. The number of vectors associated with each component within a state is then used to estimate the mixture weights (Jurafsky & Martin, 2007).

Thus, taking the single Gaussian phone system above, the mixtures were incremented by a factor of two until performance degradation was seen in the system. Then at each stage re-estimating and checking recognition results was performed.

In HTK, the conversion from single Gaussian HMMs to multiple mixture component HMMS is implemented using the HHEd MU command which will increase the number of components in a mixture by a process called mixture splitting. In this method, the command works by repeatedly splitting the mixture with the largest mixture weight until the required number of components is obtained.

4.10 Experimental Result

Having developed the phone HMM models, testing the performance of the resulting systems was done using the tool HVite which implements the Viterbi decoding algorithm. The decoding process is controlled by a recognition network compiled from a word-level network, a dictionary and a set of trained HMMs. The following command was executed to recognize the set of parameterized speech files in testscript.txt to the word level transcription result.mlf.

```
HVite -H HMM/hmm15/macros -H HMM/hmm15/hmm -S scripts/test/testscript.txt -l '*'
-i result/result.mlf -w lm/lm2 -p 19 -s 6 phones/tphones.txt
```

The options -p and -s set the word insertion penalty and the grammar scale factor, respectively. These parameters have a significant effect on recognition performance and hence, a series of tests were done on the development test data which showed –p 19 –s 6 resulted in better recognition performance.

However, the Hvite tool only supports bigram language model types and is not able to support trigram language models. Thus for experiments which use trigram language models instead of bigram models, the following command is run:

```
HDecode -o ST -H HMM/hmm15/hmm –S Script/testscript.txt -t 220.0 220.0 –C
Configs/config.hdecode -i Result/result.mlf –w LM/lm3_dio -p 100.0 -s 1.0
/Dictionary/testdictionary.txt /Triphone/tiedout.list
```

The transcriptions output by the recognizer result.mlf were compared against the MLF testword.mlf containing the correct reference word level transcription for each test utterance using the HTK analysis tool HResults. This tool uses a dynamic programming-based string alignment procedure. The following command was executed to evaluate the performance of the system:

```
HResults -I MLF/testword.mlf Phones/tphones.txt Result/result.mlf
```

4.10.1 Baseline Tigrinya Recognizer

As a baseline recognizer, 600 Tigrinya sentences were used to train the 40 (including sil) monophone HMM models and later on the tri-phone models, no Amharic data was used to train the models in this experiment. All the training data was pure Tigrinya and 300 Tigrinya sentences were used to test the models. Different Gaussian mixtures were tested starting with a single mixture and incrementing by two until the performance of the recognizer no longer increased. The experiment was done both on a bigram and a trigram language model. For the trigram language model the mixture which resulted in the best (Bigram) result was used. Table 4.2 shows the results obtained on a bigram and trigram language models.

Gaussian Mixtures	Correctly Recognized Words (%)	Accuracy
Monophone		
1	57.19%	40.04%
Triphone		
2	70.14%	48.46%
4	73.49%	55.40%
6	73.99%	60.69%
8	74.06%	61.34%
10	76.19%	64.39%
12	**80.80%**	**67.39%**
14	77.00%	64.87%
16	75.34%	58.79%
18	76.49%	56.80%
Trigram		
12	60.50%	32.76%

Table 4. 2 Baseline recognizer.

While using only the Tigrinya data for training, the best obtained result was with the triphone model using 12 Gaussian mixtures and tested on a bigram LM. Using this model, correctly recognized words of 80.80% and an accuracy of 67.39 % was obtained. While the monophone models obtained correctly recognized words of 57.19% and an accuracy of 40.04%. On the other hand, the tri-phone system with 12 Gaussian mixtures tested with a trigram LM obtained correctly recognized words of 60.50% and an accuracy of 32.76% which is less than that of the one obtained with the bigram language model. Even if the same dataset was used, the performance decrease in the trigram experiment can be attributed to the fact that there is less number of trigrams than bigrams in the LM.

4.10.2 Experiment 1

The first experiment was done using only Amharic data to train the 37 monophone HMM models and later on the tri-phone models. Since only Amharic data was used, the 2 phonemes found only in Tigrinya were not trained. The total amount of data used for training the recognizer consisted of 10,850 Amharic sentences. The models were trained following the same training steps described in the previous section. The training pronunciation dictionary consisted of only Amharic words whereas the decoding dictionary consisted of Tigrinya words from the test dataset. Once the models were developed using the Amharic data, 300 Tigrinya sentences were used to test its performance. Bigram and trigram language models were used in order to test this experiment. Different Gaussian mixtures were tested starting with a single mixture and incrementing by two until the performance of the recognizer no longer increased. Table 4.3 presents summary of experimental result.

Gaussian Mixtures	Correctly Recognized Words (%)	Accuracy
Monophone		
1	18.99%	14.22%
Triphone		
2	9.88%	7.61%
4	10.07%	8.00%
6	11.01%	9.24%
8	11.14%	9.88%
10	12.19%	10.63%
12	**13.74%**	**11.44%**
14	12.28%	10.22%
16	10.06%	8.03%
18	9.47%	7.11%
Trigram		
12	10.26%	8.45%

Table 4. 3 Amharic trained recognizer (1).

According to the results shown above, the highest result obtained from the experiments conducted using different parameters were the monophone models which obtained correctly recognized words of 18.99% and an accuracy of 14.22%. However, there was a decrease in performance with the tri-phone models as the best result obtained was correctly recognized words of 13.74% and an accuracy of 11.44% with 12 gaussian mixtures. The 2 untrained Tigrinya phonemes have also contributed to the decrease in performance of this experiment.

4.10.3 Experiment 2

In the second experiment, the data used to train the 39 monophone models and the tri-phones was 10,850 Amharic sentences. The 2 unique Tigrinya phones in the test data were replaced with phones from the Amharic language which were closer to them. The test pronunciation dictionary consisted of Tigrinya words out of which the phonemes in words containing the 2 unique Tigrinya phones were replaced with phones from Amharic

which were closest to them. Results were tested on a bigram and trigram LM. Table 4.4 presents summary of experimental result.

Gaussian Mixtures	Correctly Recognized Words (%)	Accuracy
Monophone		
1	19.04%	14.75%
Triphone		
2	10.90%	8.46%
4	10.95%	8.90%
6	11.95%	9.24%
8	12.14%	9.88%
10	13.14%	10.49%
12	**14.84%**	**12.44%**
14	14.28%	11.92%
16	14.00%	11.09%
18	13.47%	10.81%
Trigram		
12	12.26%	10.45%

Table 4. 4 Amharic trained recognizer (2).

While replacing the 2 unique phonemes in the test data, the highest obtained result was with the tri-phone model using 12 Gaussian mixtures and a bigram LM. Thus, correctly recognized words of 14.84% and an accuracy of 12.44 % were obtained. While the monophone models obtained correctly recognized words of 19.04% and an accuracy of 14.75%. The best system (a system with 12 Gaussian mixtures) was tested using a trigram language model and obtained correctly recognized words of 12.26% and an accuracy of 10.45 %.

4.10.4 Experiment 3

In the third experiment, the data used to train the 39 monophone models and the tri-phones was 10,850 Amharic sentences as well as 600 Tigrinya sentences. The Tigrinya data was used for adaptation as well as training of the 2 unique phones. The training pronunciation dictionary consisted of Amharic and Tigrinya words whereas the decoding dictionary consisted of Tigrinya words from the test dataset. First a set of monophone seed models were generated using only the Amharic data. This was achieved by training the models until the 8th iteration using only the Amharic data. Next, since the seed models needed to be adapted to the Tigrinya language, the 600 Tigrinya sentences were used to retrain the seed models until better estimates could be achieved. Results were tested on a bigram and trigram LM. Table 4.5 presents summary of experimental result.

Gaussian Mixtures	Correctly Recognized Words (%)	Accuracy
Monophone		
1	59.16%	42.22%
Triphone		
2	77.19%	60.69%
4	79.22%	62.11%
6	80.09%	65.43%
8	83.00%	66.80%
10	85.90%	70.52%
12	**88.33%**	**73.43%**
14	86.20%	72.14%
16	83.33%	71.15%
18	84.25%	72.66%
Trigram		
12	74.29%	39.19%

Table 4. 5 Amharic and Tigrinya trained recognizer (1).

While using mixture of Amharic and Tigrinya data for training, the highest obtained result was with the tri-phone model using 12 Gaussian mixtures and a bigram LM. Thus, correctly recognized words of 88.33% and an accuracy of 73.43 % were obtained. While the monophone models obtained correctly recognized words of 59.16% and an accuracy of 42.22%. The best system (a system with 12 Gaussian mixtures) was tested using a trigram language model and obtained correctly recognized words of 74.29% and an accuracy of 39.19 %.

4.10.5 Experiment 4

In the fourth experiment, the data used to train the 39 monophone models as well as the tri-phones was the same as that of the data used in experiment 3. The difference between these two experiments lies in the way the models were trained. In this experiment the models were trained starting from the first iteration using both the Amharic and Tigrinya data. The models were tested on a bigram and trigram LM. Table 4.6 shows the experimental results obtained.

Gaussian Mixtures	Correctly Recognized Words (%)	Accuracy
Monophone		
1	13.04%	9.00%
Triphone		
2	11.22%	7.98%
4	13.80%	9.22%
6	14.09%	9.89%
8	15.37%	10.02%
10	17.00%	12.35%
12	**19.85%**	**13.01%**
14	18.55%	12.98%
16	13.05%	17.28%
18	17.28%	12.57%
Trigram		
12	14.07%	9.11%

Table 4. 6 Amharic and Tigrinya trained recognizer (2).

As compared to experiment 3, there is a major decrease in the results obtained here. A percentage of correctly recognized words of 19.85% and an accuracy of 13.01 % were obtained. While the monophone models obtained correctly recognized words % of 13.04 and an accuracy of 9.00 %.

4.10.6 Experiment 5

The fifth experiment was done using the same parameters as that of experiment 3. The change in this experiment was in the model topology used. All the previous experiments were done using a 3 emitting state left to right HMM model topology. This experiment used a 5 emitting state left to right topology for the models to see if performance could be improved. The models were tested on a bigram LM and a trigram LM. Table 4.7 presents summary of experimental result.

Gaussian Mixtures	Correctly Recognized Words (%)	Accuracy
Monophone		
1	27.67%	13.44%
Triphone		
2	22.27%	15.68%
4	23.98%	18.05%
6	25.56%	18.05%
8	24.11%	18.58%
10	25.16%	19.10%
12	**25.82%**	**19.37%**
14	25.16%	19.24%
16	24.51%	18.18%
18	15.03%	18.05%
Trigram		
12	27.67%	13.44%

Table 4. 7 Five emitting state topology recognizer.

4.11 Analysis of the Result

The first experiment was done using only Amharic data to train the 37 phone HMM models. Both monophone and tri-phone models were developed in this experiment. The highest result obtained from the two was the monophone recognizer which obtained correctly recognized words of 18.99% and an accuracy of 14.22%. Whereas the tri-phone system showed a decrease in performance with correctly recognized words of 13.74% and an accuracy of 11.44% with 12 Gaussian mixtures. The decrease in performance with the tri-phone system resulted due to the tri-phone model, which has a wider context causing the problem of phonetic context mismatch between Amharic and Tigrinya language. Words consisting of the unique Tigrinya phonemes haven't been handled in this experiment. Which shows that small amount of Tigrinya data is needed and also the influence and significance of the unique untrained phonemes for the performance of the recognizer.

The second experiment, the highest result obtained was correctly recognized words of 14.84% and an accuracy of 12.44 %. Replacing the 2 unique Tigrinya phones in the test data with the closest Amharic phones yielded in no performance improvement. This could be attributed to the assumption made in the similarity between the 2 unique Tigrinya phones and the ones chosen from the Amharic phones.

The third experiment obtained the best result from all the experiments conducted. Using a mixture of the 10,850 Amharic sentences and the 600 Tigrinya sentences for training the recognizer, the highest obtained result was with the tri-phone model using 12 Gaussian mixtures and a bigram LM. This recognizer achieved correctly recognized words of 88.33% and an accuracy of 73.43 %. Better results were achieved in this experiment because limited amount of Tigrinya data was used to adapt the Amharic seed models to the Tigrinya language as well as the training the 2 unique phonemes.

In the fourth experiment there was a major decrease in the results obtained as compared to the third experiment. Correctly recognized words of 19.85% and an accuracy of 13.14

% were obtained with the triphone model, while the monophone models obtained correctly recognized words of 13.04% and an accuracy of 9.00 %. Trainings in both the third and fourth experiments were done using the same set of data. The difference was in the way models were trained. In the fourth experiment the models were trained starting from the first iteration using both the Amharic and Tigrinya data whereas in the third one, the Tigrinya data was used after the 8^{th} iteration of the training process. The major decrease in the performance obtained in the fourth experiment is attributed to the way the models were trained. The results obtained here show that using the Tigrinya data for adaptation is not enough and that the method used in training the models also affects performance.

The experiment done using a 5 emitting state left to right topology obtained a correctly recognized words of 25.82% and an accuracy of 19.37 % with the tri-phone system using bigram language model. Whereas the trigram language model obtained correctly recognized words of 15.42% and an accuracy of 11.33%. Changing the state from 3 to 5 with the intent of increasing performance didn't obtain better result, in fact it resulted in a performance decrease. This shows that it is better to use 3 emitting states topology for both the monophone and tri-phone system.

The baseline Tigrinya recognizer which was trained using only the 600 Tigrinya data resulted in correctly recognized words of 57.19% and an accuracy of 40.04 % with the monophone model and correctly recognized words of 80.80% and an accuracy of 67.39 % with the tri-phone model.

In the previous research by (Hafte, 2009) for speaker independent Tigrinya ASR trained on 300 sentences, he obtained 60.32% correctly recognized words and an accuracy of 58.38 %. Comparing this result and the result of the baseline model to the best obtained result in this research which was correctly recognized words of 88.33% and an accuracy of 73.43 % shows that considerable improvement has been achieved in performance improvement of Tigrinya ASR.

CHAPTER FIVE

Conclusion and Recommendation

5.1 Introduction

This section presents the conclusions made from the experiments conducted and future research directions that can be taken in this area as an extension of this work to improve Tigrinya LVCSR systems.

5.2 Conclusion

Automatic Speech Recognition technology has made it possible for computers to follow human voice commands and understand human languages with the main aim of developing techniques and systems for speech input to machines. The past few years has seen tremendous success for automatic speech recognition (ASR) in several different languages. However most of the approaches used in ASR development are statistical, which mainly rely on the availability of large amounts of speech and text data. The rapid transfer of ASR technologies to new languages is hampered by the non-availability of sufficient data in the new languages. Speech developers need to cope with very little or no data, typically obtained from a different target domain.

This study has introduced approaches that allow the development of LVCSR systems in a new target language without the need of large speech databases in that language. A Tigrinya LVCSR was developed using an Amharic Corpus and a limited Tigrinya data to train the acoustic models. The study was conducted based on the assumption that the articulatory representations of phonemes are similar across the Tigrinya and Amharic languages with the exception of the 2 phonemes unique to Tigrinya and using all the phonemes as acoustic units.

Several experiments were conducted in this study. The experiment done using only Amharic data to train the 37 phone HMM models obtained a correctly recognized words of 18.99 and an accuracy of 14.22% for the monophone model and correctly recognized words of 13.74% and an accuracy of 11.44% with 12 Gaussian mixtures for the tri-phone model. The decrease in performance with the tri-phone system was attributed to a wider context raising the problem of phonetic context mismatch between Amharic and Tigrinya language. Words consisting of the unique Tigrinya phonemes haven't been handed in this experiment.

The experiment conducted using a mixture of the 10,850 Amharic sentences and the 600 Tigrinya sentences obtained the highest result with the tri-phone model using 12 Gaussian mixtures of correctly recognized words of 88.33% and an accuracy of 73.43 %. Better results were achieved in this experiment because limited amount of Tigrinya data was used to adapt the Amharic seed models to the Tigrinya language and the 2 unique Tigrinya phonemes were also trained, which means words consisting of the unique phonemes were handled. To further increase performance, the other two experiments were done by changing the way that the models were trained and changing the number of the state of the models. This showed a major decrease in performance by obtaining correctly recognized words of 19.85% and an accuracy of 13.01 % with tri-phone model for the experiment done by changing the training method and correctly recognized words of 25.82% and an accuracy of 19.37 % tri-phone by changing the topology.

On the other hand, the baseline Tigrinya recognizer which was trained using only the 600 Tigrinya data resulted in correctly recognized words of 57.19% and an accuracy of 40.04 % with the monophone model and correctly recognized words of 80.80% and an accuracy of 67.39 % on the tri-phone model with 12 Gaussian mixtures. Comparing this result with the best result obtained in the experiment showed that an increase of about 8% was achieved in terms of correctly recognized words and of about 6% in terms of accuracy. This has proven that the use of Amharic data with limited Tigrinya data for training a Tigrinya recognizer does result in significant performance increase and that it is a promising future research direction given that different methods are applied to further achieve better results.

5.3 Recommendations

Based on the processes implemented, the results obtained and the different problems encountered in this research, the following recommendations are forwarded.

- The Tigrinya speech recognizer designed in this experiment was trained using Amharic data as a source language and limited Tigrinya data, is only able to recognize Tigrinya data. The aim of this was to counter balance the sparseness of the Tigrinya data by using resources available from the Amharic language. This recognizer can be further expanded into a bilingual LVCSR system by enabling it to also recognize Amharic speech data with the addition of a bilingual pronunciation dictionary.
- This Research was done based on IPA assumption that the articulatory representations of phonemes are so similar across languages that they can be considered as units that are independent from the underlying language and thus assuming that most of the phones are similar between Amharic and Tigrinya. Further building on what has already been done to find ways of handling the 2 unique phonemes. As this is the first attempt other phone mapping techniques and approaches such as the data driven approach can also be tried for performance improvement purpose.
- This research can also be conducted for other under resourced local languages which have common phonemes among them and for which no Speech Recognizer has been developed or which obtained poor results due to limited amount of data.
- The pronunciation dictionary used does not handle the problem of gemination of consonants and the irregular realization of the sixth order vowel and the glottal stop consonant. The construction of a pronunciation dictionary which handles these problems is important.
- A detailed knowledge based construction of the file used to perform the parameter tying is needed while training tri-phone models.
- Language and speaker adaptation techniques can also increase the performance of the recognizer.

References

Adami, A. (2010). Automatic Speech Recognition: From the Beginning to the Portuguese Language. *9th International Conference on Computational Processing of the Portuguese Language*.

Adey, E. (2013). *Investigating the use of large syllable acoustic units for Amharic Automatic Speech Recognition.* Addis Ababa University. Addis Ababa, Ethiopia: Unbublished.

Ager, S. (2015). *www.omniglot.com*. Retrieved April 14, 2014, from http://www.omniglot.com/writing/ipa.html

Ai-jun, L., & Zhi-gang, Y. (2006). Standardization Of Speech Corpus. *Data Science Journal, 6*, 806-811.

Andreas, S. (2002). SRILM —An Etensible Language Modeling Toolkit. *In Proceedings of the 7th International conference on spoken language processing, 2*, 901-904.

Anusuya, M., & Katti, S. (2009). Speech Recognition by Machine: A Review. *International Jornal of Computer Science and Information Security, 6*(3), 181-205.

Baye, Y. (2008). *የአማርኛ ሰዋሰው የተሻሻለ ሁለተኛ ዕትም.* Addis Ababa.

Blunsom, P. (2004). Hidden Markov Models.

Brinton, L. J. (2000). *The Structure of Modern English.* Amsterdam: John Benjamins Publishing Co.

Daniel, P. T. (1997). Scripts of Semetic Languages. In R. Hetzron, *The Semitic Languages.* New York: Routledge.

Das, S. (2012). Speech Recognition Technique: A Review. *International Journal of Engineering Research and Applications, 2*(3), 2071-2087.

Hafte, A. (2009). *Hidden Markov Model Based Tigrigna Speech Recognition.* Addis Ababa University. Addis Ababa, Ethiopia: Unpublished.

Ilya, O. (2006). Statistical Language Models for Dialogue Systems.

Ilya, O. (2008). *Language models for automatic speech recognition of inflectional languages.* PhD Thesis, University of West Bohemia, Computer Science and Engineering, West Bohemia.

Juang, B., & Rabiner, L. R. (2004). Automatic Speech Recognition – A Brief History of the Technology Development.

Jurafsky, D., & Martin, J. H. (2007). *Speech and Language Processing: An introduction to natural language processing* (2nd ed.). Prentice Hall.

Kesarkar, M. P. (2003). Feature Extraction for Speech Recognition. *Electronic system group*.

Kinfe, T. (2002). *Sub-Word Based Amharic Word Recognition : An Experiment Using Hidden Markov Model (HMM).* Addis Ababa University. Addis Ababa, Ethiopia: Unpublished.

Kogan, L. E. (1997). The Semitic Languages. In R. Hetzron, *The Semitic Languages.* New York: Routledge.

Lee, K.-F. (1989). *Automatic Speech Recognition The Development of the SPHINX System.* Boston: Kluwer Acadamic Publishers.

Leslau, W. (2000). *Introductory Grammar of Amharic.* Harrassowits Verlag.

Liu, C., & Melnar, L. (2006). Training Acoustic Models with Speech Data from Different Languages. *Human Interaction Research, Motorola*.

Martin, A.-D., Lamel, L., & Adda, G. (2014). A first LVCSR system for Luxemburgish, an under-resourced European Language. *Springer International Publishing*, 479–490.

Naveen , S., & Shrikanth, N. (2003). Language-Adaptive Persian Speech Recognition. *In Proc. Eurospeech*.

Peinado, A., & Segura, J. (2006). *Speech Recognition Over Digital Channels: Robustness and Standards.* ISBN.

Phillips, J. (2013, May 13). *aboutworldlanguages.com*. Retrieved February 4, 2015, from http://aboutworldlanguages.com/amharic

Pires, N. J. (2007). *Industrial Robots Programming.* New York: Springer Science & Business Media.

Rabiner, L., & Juang, B.-H. (1993). *Fundamentals of Speech Recognition.* NJ, Englewood Cliffs: Prentice Hall Inc.

Rahul , C., Venkatesh, K., Anumanchipalli, G., & Joshi, S. (2005). Lexical Modeling for Non Native Speech Recognition using Neural Networks. *In Proceedings of the International Conference on Natural Language Processing*.

Sharma, G. (2013). English in Ethiopia. *STAR Journal*, 74-85.

Shimelis, M. (2014). *Nominalization via verbal Derivations Amharic, Tigrinya and oromo.* Addis Ababa University. Addis Ababa, Ethiopia: Unpublished.

Shu-qin, Z., Shao-qian, W., & Yin-xia, Z. (2012). *Research and Analysis of HMM Speech Recognition Model* (Vol. 12). Beijing, China: International Conference on Artificial Intelligence and Soft Computing.

Singh, B., Singh, S., Singh, J., Gobind , H. P., & Kapur, N. (2012). Hidden Markov Model Based Interface to Control Computer Machine through Speech Commands for Punjabi Language. *International Journal of Computer Sci ence And Technology*.

Solomon, B. (2001). *Isolated Amharic Consonant-Vowel syllable recognition: An Experiment using the Hidden Markov Model.* Addis Ababa University. Addis Ababa, Ethiopia: Unpublished.

Solomon, T. (2006). *Automatic Speech Recognition for Amharic.* PhD Thesis, Hamburg, Germany.

Solomon, T., & Wolfgang, M. (2007). Syllable-Based Speech Recognition for Amharic. *Proceedings of the 2007 Workshop on Computational Approaches to Semitic Languages: Common Issues and Resources*.

Solomon, T., Wolfgang, M., & Bairu, T. (2005). An Amharic Speech Corpus for Large Vocabulary Continuous Speech. *INTERSPEECH*, 1601-1604.

Tanja , S., & Alex , W. (2001). Language-independent and language-adaptive acoustic modeling for speech recognition. *35*, 31 - 51.

Tesfay, T. Y. (2002). *A Modern Grammar of Tigrinya.* Rome.

Woodland, P. (2014). *htk.eng.cam.ac.uk*. Retrieved May 24, 2014, from http://htk.eng.cam.ac.uk/

Young, S., & Gales , M. (2008). *The Application of Hidden Markov Models in Speech Recognition.*

Young, S., Evermann, G., Gales, M., Hain, T., Kershaw, D., Liu, X., et al. (2009). *The HTK Book.* Cambridge University Engineering Department.

Yule, G. (1996). *The Study of Language.* UK: Cambridge University Press.

Zegaye, S. (2003). *Hidden Markov Model large vocabulary, speaker independent continous Amharic speech recognition.* Addis Ababa University. Addis Ababa, Ethiopia: Unpublished.

Appendix A: Tigrinya Graphemes

Name	IPA	e	u	i	a	ie	ɨ/-	o
pa	p	ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ
ba	b	በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ
pha	p'	ጰ	ጱ	ጲ	ጳ	ጴ	ጵ	ጶ
ma	m	መ	ሙ	ሚ	ማ	ሜ	ም	ሞ
fa	f	ፈ	ፉ	ፊ	ፋ	ፌ	ፍ	ፎ
va	v	ቨ	ቩ	ቪ	ቫ	ቬ	ቭ	ቮ
wa	w	ወ	ዉ	ዊ	ዋ	ዌ	ው	ዎ
ta	t	ተ	ቱ	ቲ	ታ	ቴ	ት	ቶ
da	d	ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ
tha	t'	ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ
tsa	ts'	ጸ	ጹ	ጺ	ጻ	ጼ	ጽ	ጾ
na	n	ነ	ኑ	ኒ	ና	ኔ	ን	ኖ
sa	s	ሰ	ሱ	ሲ	ሳ	ሴ	ስ	ሶ
za	z	ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ
ra	r	ረ	ሩ	ሪ	ራ	ሬ	ር	ሮ
la	l	ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ
ca	ʧ	ቸ	ቹ	ቺ	ቻ	ቼ	ች	ቾ
ja	ʤ	ጀ	ጁ	ጂ	ጃ	ጄ	ጅ	ጆ
cha	ʧ'	ጨ	ጩ	ጪ	ጫ	ጬ	ጭ	ጮ

Name	IPA	e	u	i	a	ie	ɨ/-	o
nya	ɲ	ኘ	ኙ	ኚ	ኛ	ኜ	ኝ	ኞ
sha	ʃ	ሸ	ሹ	ሺ	ሻ	ሼ	ሽ	ሾ
zha	ʒ	ዠ	ዡ	ዢ	ዣ	ዤ	ዥ	ዦ
ya	j	የ	ዩ	ዪ	ያ	ዬ	ይ	ዮ
ka	k	ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ
kwa	kw	ኰ		ኵ	ኳ	ኴ	ኵ	
ga	g	ገ	ጉ	ጊ	ጋ	ጌ	ግ	ጎ
gwa	gw	ጐ		ጕ	ጓ	ጔ		
qa	k'	ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ
qwa	kw'	ቈ		ቊ	ቋ	ቌ	ቍ	
kxa	x	ኸ	ኹ	ኺ	ኻ	ኼ	ኽ	ኾ
kxwa	xw	ዀ		ዂ	ዃ	ዄ	ዅ	
qha	ʁ'	ቐ	ቑ	ቒ	ቓ	ቔ	ቕ	ቖ
qhwa	ʁw	ቘ		ቚ	ቛ	ቜ	ቝ	
hha	ħ	ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ
'ä	ʕ	ዐ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ
'ä	ʔ	አ	ኡ	ኢ	ኣ	ኤ	እ	ኦ
ha	h	ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ

Appendix B: Pronunciation Dictionary

AaQEmEnE A a Q E m E n E
AaQEmii A a Q E m ii
AaQwiirenE A a Q w ii r e n E
AabEyE A a b E y E
AabEyii A a b E y ii
AabEyu A a b E y u
Aabiiyii A a b ii y ii
AadEgEratE A a d E g E r a t E
CarEterE C a r E t e r E
Eba E b a
EhhulE E hh u l E
EnEfExEmelomE E n E f E x E m e l o m E
EnEgEnEzebo E n E g E n E z e b o
EnEgiihenE E n E g ii h e n E
EnEriio E n E r ii o
EnEtESErefE E n E t E S E r e f E
HuQe H u Q e
QerebE Q e r e b E
Qereba Q e r e b a
Qeriixu Q e r ii x u
Qexalii Q e x a l ii
SEHE S E H E
SEHEnE S E H E n E
SEQaQE S E Q a Q E
SEduSEte S E d u S E t e
.
.
.
zeyEzayEdE z e y E z a y E d E
zeyedElE z e y e d E l E
zeyetegEbEru z e y e t e g E b E r u
ziega z ie g a
ziegatatE z ie g a t a t E
ziegatatEna z ie g a t a t E n a
zienawii z ie n a w ii

Appendix C: The Configuration File (Config.txt)

```
SOURCEFORMAT = WAVE
TARGETKIND = MFCC_0
TARGETRATE = 100000.0
SAVECOMPRESSED = T
SAVEWITHCRC = T
WINDOWSIZE = 250000.0
USEHAMMING = T
PREEMCOEF = 0.97
NUMCHANS = 26
CEPLIFTER = 22
NUMCEPS = 12
ENORMALISE = F
```

Appendix D: The Prototype Definition

~o
<STREAMINFO> 1 39
<VECSIZE> 39<NULLD><MFCC_D_A_0><DIAGC>
~h "proto"
<BeginHMM>
<NumStates> 5
<State> 2
<Mean> 39
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
<Variance> 39
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
<State> 3
<Mean> 39
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
<Variance> 39
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
<State> 4
<Mean> 39
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
<Variance> 39
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
<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: The tree.hed File

```
RO 100.0
TR 0
QS "R_NonBoundary"   { *+* }
QS "R_Vowel" {a+*,e+*,ii+*,E+*,Ie+*,o+*,u+*}
QS "R_Consonant"
{b+*,C+*,X+*,d+*,f+*,g+*,h+*,hh+*,j+*,k+*,l+*,m+*,n+*,N+*,q+*,P+*,p+*,r+*,s+*,S+*,t+*,
T+*,v+*,w+*,y+*,z+*,Z+*,H+*,Q+*,K+*,x+*}
QS "R_Stop"    {b+*,d+*,P+*,g+*,k+*,p+*,t+*,T+*,q+*,^e+*}
QS "R_Nasal"   {m+*,n+*,N+*}
QS "R_Fricative"          {f+*,x+*,S+*,s+*,x+*,z+*,Z+*,h+*,H+*,A+*}
QS "R_Liquid"  {l+*,r+*}
QS "R_Strident"           {Z+*,S+*,j+*,C+*,f+*}
QS "R_UnStrident"       {*b+*,*m+*,*t+*,*d+*,*n+*,*k+*,*g+*,*l+*,*r+*,*y+*,*w+* }
QS "R_Front"   {ii+*,Ie+*,P+*,f+*,m+*,p+*,v+*,w+*}
QS "R_Central"
          {a+*,e+*,E+*,d+*,T+*,S+*,l+*,n+*,N+*,r+*,s+*,t+*,H+*,z+*,Z+*,C+*,j+*,X+*,h+*,}
QS "R_Back"    {y+*,k+*,g+*,q+*,k+*,o+*,u+*}
QS "R_Fortis"        { t+*,k+*,s+*,f+*,P+*,K+*}
QS "R_Lenis"         { b+*,d+*,g+*}
QS "R_UnFortLenis"   { m+*,n+*,l+*,r+*,y+*,w+* }
QS "R_Coronal"       { t+*,d+*,n+*,s+*,l+*,r+*,N+*}
QS "R_NonCoronal"    { b+*,m+*,k+*,g+*,y+*,w+*,K+*}
QS "R_Anterior"      { b+*,m+*,t+*,d+*,n+*,s+*,l+*,w+*}
QS "R_NonAnterior"   { k+*,g+*,r+*,y+*,K+*}
QS "R_Continuent"    {
Z+*,S+*,f+*,z+*,x+*,s+*,l+*,w+*,y+*,h+*,A+*,H+*,e+*,u+*,ii+*,a+*,Ie+*,E+* ,o+*}
QS "R_NonContinuent" { b+*,t+*,d+*,k+*,g+*,K+*}
QS "R_Front_Vowel"              {ii+*,Ie+*}
QS "R_Central_Vowel"            {a+*,E+*,e+*}
QS "R_Back_Vowel"               {o+*,u+*}
QS "R_long_Vowel"               {ii+*,Ie+*}
QS "R_Front_Start_Vowel"        {aw+*,ax+*,Ie+*,ii+*}
QS "R_Fronting_Vowel"                   {ay+*,ey+*,o+*}
QS "R_High_Vowel"               {ii+*,E+*,u+*}
QS "R_Medium_Vowel"                     {Ie+*,e+*,o+*}
QS "R_low_Vowel"                {a+*}
QS "R_Rounded_Vowel"                    {o+*,u+*}
QS "R_Unrounded_Vowel"                  {a+*,e+*,Ie+*,ii+*,E+*}
QS "R_Reduced_Vowel"                    {ax+*,E+*}
QS "R_IVowel"                   {ii+*,Ie+*}
QS "R_EVowel"                           {e+*,E+*}
QS "R_AVowel"                           {a+*}
QS "R_OVowel"                           {o+*}
QS "R_UVowel"                           {u+*}
QS "R_Unvoiced_Consonant"   {f+*,k+*,p+*,s+*,S+*,t+*,T+*,P+*,q+*,e+*,x+*,h+*,H+*,}
QS "R_Voiced_Consonant"
          {b+*,d+*,g+*,l*,m+*,n+*,N+*,r+*,v+*,w+*,y+*,z+*,Z+*C+*,x+*,}
QS "R_Front_Consonant"                  {b+*,f+*,m+*,p+*,v+*,w+*,P+*}
```

```
QS "R_Central_Consonant"
        {d+*,C+*,X+*,j+*,l+*,n+*,N+*,r+*,s+*,t+*,T+*,z+*,Z+*,x+*,y+*}
QS "R_Back_Consonant"                   {g+*,k+*,y+*,q+*}
QS "R_Voiced_Stop"              {b+*,d+*,g+*}
QS "R_Unvoiced_Stop"            {p+*,t+*,k+*,q+*,P+*,T+*,e+*}
QS "R_Front_Stop"               {b+*,p+*,P+*}
QS "R_Central_Stop"             {d+*,t+*,T+*}
QS "R_Back_Stop"                {g+*,k+*,q+*}
QS "R_Voiced_Fricative"                 {z+*,A+*,Z+*}
QS "R_Unvoiced_Fricative"       {f+*,s+*,x+*,S+*,H+*,h+*}
QS "R_Front_Fricative"          {f+*,v+*}
QS "R_Central_Fricative"        {s+*,z+*x+*}
QS "R_Back_Fricative"           {S+*,Z+*}
QS "R_Affricate_Consonant"      {C+*,j+*,X+*}
QS "R_Not_Affricate"            {f+*,s+*,S+*,T+*,v+*,z+*,Z+*,x+*}
QS "R_silences"                         {sil+*}
QS "R_p"                                    {p+*}
QS "R_t"                                    {t+*}
QS "R_k"                                    {k+*}
QS "R_b"                                    {b+*}
QS "R_d"                                    {d+*}
QS "R_g"                                    {g+*}
QS "R_P"                                    {P+*}
QS "R_T"                                    {T+*}
QS "R_X"                                    {X+*}
QS "R_q"                                    {q+*}
QS "R_f"                                    {f+*}
QS "R_s"                                    {s+*}
QS "R_S"                                    {S+*}
QS "R_h"                                    {h+*}
QS "R_hh"                                   {hh+*}
QS "R_H"                                    {H+*}
QS "R_C"                                    {c+*}
QS "R_j"                                    {j+*}
QS "R_m"                                  {m+*}
QS "R_n"                                    {n+*}
QS "R_N"                                    {N+*}
QS "R_l"                                    {l+*}
QS "R_r"                                    {r+*}
QS "R_y"                                    {y+*}
QS "R_w"                                    {w+*}
QS "R_v"                                    {v+*}
QS "R_z"                                    {z+*}
QS "R_Z"                                    {Z+*}
QS "R_e"                                    {e+*}
QS "R_u"                                    {u+*}
QS "R_ii"                                   {ii+*}
QS "R_a"                                    {a+*}
QS "R_Ie"                                 {Ie+*}
QS "R_E"                                    {E+*}
QS "R_o"                                    {o+*}
```

```
QS "R_Q"                                {Q+*}
QS "R_sil"                              {sil+*}
QS "R_K"                                {K+*}
QS "R_A"                                {A+*}
QS "L_NonBoundary"   { *-* }
QS "L_Vowel" {a-*,e-*,ii-*,E-*,Ie-*,o-*,u-*}
QS "L_Consonant" {b-*,C-*,X-*,d-*,f-*,g-*,h-*,j-*,k-*,l-*,m-*,n-*,N-*,q-*,P-*,p-*,r-*,s-*,S-*,t-
*,T-*,v-*,w-*,y-*,z-*,Z-*,H-*,Q-*,K-*,x-*}
QS "L_Stop"    {b-*,d-*,P-*,g-*,k-*,p-*,t-*,T-*,q-*,^e-*}
QS "L_Nasal"   {m-*,n-*,N-*}
QS "L_Fricative"        {f-*,x-*,S-*,s-*,x-*,z-*,Z-*,h-*,H-*,A-*}
QS "L_Liquid"  {l-*,r-*}
QS "L_Strident"         {Z-*,S-*,j-*,C-*,f-*}
QS "L_UnStrident"       {*b-*,*m-*,*t-*,*d-*,*n-*,*k-*,*g-*,*l-*,*r-*,*y-*,*w-* }
QS "L_Front"   {ii-*,Ie-*,P-*,f-*,m-*,p-*,v-*,w-*}
QS "L_Central" {a-*,e-*,E-*,d-*,T-*,S-*,l-*,n-*,N-*,r-*,s-*,t-*,H-*,z-*,Z-*,C-*,j-*,X-*,h-*,}
QS "L_Back"    {y-*,k-*,g-*,q-*,k-*,o-*,u-*}
QS "L_Lenis"        { b-*,d-*,g-*}
QS "L_UnFortLenis"   { m-*,n-*,l-*,r-*,y-*,w-* }
QS "L_Coronal"       { t-*,d-*,n-*,s-*,l-*,r-*,N-*}
QS "L_NonCoronal"    { b-*,m-*,k-*,g-*,y-*,w-*,K-*}
QS "L_Anterior"      { b-*,m-*,t-*,d-*,n-*,s-*,l-*,w-*}
QS "L_NonAnterior"   { k-*,g-*,r-*,y-*,K-*}
QS "L_Continuent"    { Z-*,S-*,f-*,z-*,x-*,s-*,l-*,w-*,y-*,h-*,A-*,H-*,e-*,u-*,ii-*,a-*,Ie-*,E-*
,o-*}
QS "L_NonContinuent" { b-*,t-*,d-*,k-*,g-*,K-*}
QS "L_Front_Vowel"              {ii-*,Ie-*}
QS "L_Central_Vowel"            {a-*,E-*,e-*}
QS "L_Back_Vowel"               {o-*,u-*}
QS "L_long_Vowel"               {ii-*,Ie-*}
QS "L_Front_Start_Vowel"        {aw-*,ax-*,Ie-*,ii-*}
QS "L_Fronting_Vowel"                   {ay-*,ey-*,o-*}
QS "L_High_Vowel"               {ii-*,E-*,u-*}
QS "L_Medium_Vowel"                     {Ie-*,e-*,o-*}
QS "L_low_Vowel"                         {a-*}
QS "L_Rounded_Vowel"                    {o-*,u-*}
QS "L_Unrounded_Vowel"                  {a-*,e-*,Ie-*,ii-*,E-*}
QS "L_Reduced_Vowel"                    {ax-*,E-*}
QS "L_IVowel"                   {ii-*,Ie-*}
QS "L_EVowel"                           {e-*,E-*}
QS "L_AVowel"                           {a-*}
QS "L_OVowel"                           {o-*}
QS "L_UVowel"                           {u-*}
QS "L_Unvoiced_Consonant"   {f-*,k-*,p-*,s-*,S-*,t-*,T-*,P-*,q-*,e-*,x-*,h-*,H-*,}
QS "L_Voiced_Consonant"                 {b-*,d-*,g-*,l*,m-*,n-*,N-*,r-*,v-*,w-*,y-*,z-*,Z-*C-
*,x-*,}
QS "L_Front_Consonant"                  {b-*,f-*,m-*,p-*,v-*,w-*,P-*}
QS "L_Central_Consonant"        {d-*,C-*,X-*,j-*,l-*,n-*,N-*,r-*,s-*,t-*,T-*,z-*,Z-*,x-*,y-*}
QS "L_Back_Consonant"                   {g-*,k-*,y-*,q-*}
QS "L_Fortis"                 {s-*,k-*,t-*,}
QS "L_Voiced_Stop"              {b-*,d-*,g-*}
```

```
QS "L_Unvoiced_Stop"          {p-*,t-*,k-*,q-*,P-*,T-*,e-*}
QS "L_Front_Stop"             {b-*,p-*,P-*}
QS "L_Central_Stop"           {d-*,t-*,T-*}
QS "L_Back_Stop"              {g-*,k-*,q-*}
QS "L_Voiced_Fricative"                {z-*,A-*,Z-*}
QS "L_Unvoiced_Fricative"     {f-*,s-*,x-*,S-*,H-*,h-*}
QS "L_Front_Fricative"        {f-*,v-*}
QS "L_Central_Fricative"      {s-*,z-*x-*}
QS "L_Back_Fricative" {S-*,Z-*}
QS "L_Affricate_Consonant"    {C-*,j-*,X-*}
QS "L_Not_Affricate"  {f-*,s-*,S-*,T-*,v-*,z-*,Z-*,x-*}
QS "L_silences"                        {sil-*}
QS "L_p"                                  {p-*}
QS "L_t"                                  {t-*}
QS "L_k"                                  {k-*}
QS "L_b"                                  {b-*}
QS "L_d"                                  {d-*}
QS "L_g"                                  {g-*}
QS "L_P"                                  {P-*}
QS "L_T"                                  {T-*}
QS "L_X"                                  {X-*}
QS "L_q"                                  {q-*}
QS "L_f"                                  {f-*}
QS "L_s"                                  {s-*}
QS "L_S"                                  {S-*}
QS "L_h"                                  {h-*}
QS "L_H"                                  {H-*}
QS "L_C"                                  {c-*}
QS "L_j"                                  {j-*}
QS "L_m"                                 {m-*}
QS "L_n"                                  {n-*}
QS "L_N"                                  {N-*}
QS "L_l"                                  {l-*}
QS "L_r"                                  {r-*}
QS "L_y"                                  {y-*}
QS "L_w"                                  {w-*}
QS "L_v"                                  {v-*}
QS "L_z"                                  {z-*}
QS "L_Z"                                  {Z-*}
QS "L_e"                                  {e-*}
QS "L_u"                                  {u-*}
QS "L_ii"                                 {ii-*}
QS "L_a"                                  {a-*}
QS "L_Ie"                                {Ie-*}
QS "L_E"                                  {E-*}
QS "L_o"                                  {o-*}
QS "L_Q"                                 {Q-*}
QS "L_sil"                              {sil_*}
QS "L_K"                                  {K-*}
QS "L_A"                                  {A-*}
```

TR 2

TB 350 "ST_A_2_" {("A","*-A+*","A+*","*-A").state[2]}
TB 350 "ST_Z_2_" {("Z","*-Z+*","Z+*","*-Z").state[2]}
TB 350 "ST_E_2_" {("E","*-E+*","E+*","*-E").state[2]}
TB 350 "ST_S_2_" {("S","*-S+*","S+*","*-S").state[2]}
TB 350 "ST_l_2_" {("l","*-l+*","l+*","*-l").state[2]}
TB 350 "ST_n_2_" {("n","*-n+*","n+*","*-n").state[2]}
TB 350 "ST_e_2_" {("e","*-e+*","e+*","*-e").state[2]}
TB 350 "ST_t_2_" {("t","*-t+*","t+*","*-t").state[2]}
TB 350 "ST_u_2_" {("u","*-u+*","u+*","*-u").state[2]}
TB 350 "ST_d_2_" {("d","*-d+*","d+*","*-d").state[2]}
TB 350 "ST_o_2_" {("o","*-o+*","o+*","*-o").state[2]}
TB 350 "ST_b_2_" {("b","*-b+*","b+*","*-b").state[2]}
TB 350 "ST_a_2_" {("a","*-a+*","a+*","*-a").state[2]}
TB 350 "ST_y_2_" {("y","*-y+*","y+*","*-y").state[2]}
TB 350 "ST_N_2_" {("N","*-N+*","N+*","*-N").state[2]}
TB 350 "ST_m_2_" {("m","*-m+*","m+*","*-m").state[2]}
TB 350 "ST_g_2_" {("g","*-g+*","g+*","*-g").state[2]}
TB 350 "ST_r_2_" {("r","*-r+*","r+*","*-r").state[2]}
TB 350 "ST_w_2_" {("w","*-w+*","w+*","*-w").state[2]}
TB 350 "ST_q_2_" {("q","*-q+*","q+*","*-q").state[2]}
TB 350 "ST_f_2_" {("f","*-f+*","f+*","*-f").state[2]}
TB 350 "ST_ii_2_" {("ii","*-ii+*","ii+*","*-ii").state[2]}
TB 350 "ST_h_2_" {("h","*-h+*","h+*","*-h").state[2]}
TB 350 "ST_hh_2_" {("hh","*-hh+*","hh+*","*-hh").state[2]}
TB 350 "ST_j_2_" {("j","*-j+*","j+*","*-j").state[2]}
TB 350 "ST_k_2_" {("k","*-k+*","k+*","*-k").state[2]}
TB 350 "ST_s_2_" {("s","*-s+*","s+*","*-s").state[2]}
TB 350 "ST_x_2_" {("x","*-x+*","x+*","*-x").state[2]}
TB 350 "ST_C_2_" {("C","*-C+*","C+*","*-C").state[2]}
TB 350 "ST_K_2_" {("K","*-K+*","K+*","*-K").state[2]}
TB 350 "ST_X_2_" {("X","*-X+*","X+*","*-X").state[2]}
TB 350 "ST_Wa_2_" {("Wa","*-Wa+*","Wa+*","*-Wa").state[2]}
TB 350 "ST_z_2_" {("z","*-z+*","z+*","*-z").state[2]}
TB 350 "ST_H_2_" {("H","*-H+*","H+*","*-H").state[2]}
TB 350 "ST_T_2_" {("T","*-T+*","T+*","*-T").state[2]}
TB 350 "ST_ie_2_" {("ie","*-ie+*","ie+*","*-ie").state[2]}
TB 350 "ST_Ie_2_" {("Ie","*-Ie+*","Ie+*","*-Ie").state[2]}
TB 350 "ST_p_2_" {("p","*-p+*","p+*","*-p").state[2]}
TB 350 "ST_P_2_" {("P","*-P+*","P+*","*-P").state[2]}
TB 350 "ST_sil_2_" {("sil","*-sil+*","sil+*","*-sil").state[2]}
TB 350 "ST_Q_2_" {("Q","*-Q+*","Q+*","*-Q").state[2]}
TB 350 "ST_W_2_" {("W","*-W+*","W+*","*-W").state[2]}
TB 350 "ST_i_2_" {("i","*-i+*","i+*","*-i").state[2]}
TB 350 "ST_O_2_" {("O","*-O+*","O+*","*-O").state[2]}
TB 350 "ST_v_2_" {("v","*-v+*","v+*","*-v").state[2]}
TB 350 "ST_wa_2_" {("wa","*-wa+*","wa+*","*-wa").state[2]}
TB 350 "ST_A_3_" {("A","*-A+*","A+*","*-A").state[3]}
TB 350 "ST_E_3_" {("E","*-E+*","E+*","*-E").state[3]}
TB 350 "ST_Z_3_" {("Z","*-Z+*","Z+*","*-Z").state[3]}

TB 350 "ST_S_3_" {("S","*-S+*","S+*","*-S").state[3]}
TB 350 "ST_l_3_" {("l","*-l+*","l+*","*-l").state[3]}
TB 350 "ST_n_3_" {("n","*-n+*","n+*","*-n").state[3]}
TB 350 "ST_e_3_" {("e","*-e+*","e+*","*-e").state[3]}
TB 350 "ST_t_3_" {("t","*-t+*","t+*","*-t").state[3]}
TB 350 "ST_u_3_" {("u","*-u+*","u+*","*-u").state[3]}
TB 350 "ST_d_3_" {("d","*-d+*","d+*","*-d").state[3]}
TB 350 "ST_O_3_" {("O","*-O+*","O+*","*-O").state[3]}
TB 350 "ST_o_3_" {("o","*-o+*","o+*","*-o").state[3]}
TB 350 "ST_b_3_" {("b","*-b+*","b+*","*-b").state[3]}
TB 350 "ST_a_3_" {("a","*-a+*","a+*","*-a").state[3]}
TB 350 "ST_y_3_" {("y","*-y+*","y+*","*-y").state[3]}
TB 350 "ST_N_3_" {("N","*-N+*","N+*","*-N").state[3]}
TB 350 "ST_m_3_" {("m","*-m+*","m+*","*-m").state[3]}
TB 350 "ST_g_3_" {("g","*-g+*","g+*","*-g").state[3]}
TB 350 "ST_r_3_" {("r","*-r+*","r+*","*-r").state[3]}
TB 350 "ST_w_3_" {("w","*-w+*","w+*","*-w").state[3]}
TB 350 "ST_q_3_" {("q","*-q+*","q+*","*-q").state[3]}
TB 350 "ST_f_3_" {("f","*-f+*","f+*","*-f").state[3]}
TB 350 "ST_ii_3_" {("ii","*-ii+*","ii+*","*-ii").state[3]}
TB 350 "ST_h_3_" {("h","*-h+*","h+*","*-h").state[3]}
TB 350 "ST_hh_3_" {("hh","*-hh+*","hh+*","*-hh").state[3]}
TB 350 "ST_j_3_" {("j","*-j+*","j+*","*-j").state[3]}
TB 350 "ST_k_3_" {("k","*-k+*","k+*","*-k").state[3]}
TB 350 "ST_s_3_" {("s","*-s+*","s+*","*-s").state[3]}
TB 350 "ST_x_3_" {("x","*-x+*","x+*","*-x").state[3]}
TB 350 "ST_C_3_" {("C","*-C+*","C+*","*-C").state[3]}
TB 350 "ST_K_3_" {("K","*-K+*","K+*","*-K").state[3]}
TB 350 "ST_X_3_" {("X","*-X+*","X+*","*-X").state[3]}
TB 350 "ST_Wa_3_" {("Wa","*-Wa+*","Wa+*","*-Wa").state[3]}
TB 350 "ST_z_3_" {("z","*-z+*","z+*","*-z").state[3]}
TB 350 "ST_H_3_" {("H","*-H+*","H+*","*-H").state[3]}
TB 350 "ST_T_3_" {("T","*-T+*","T+*","*-T").state[3]}
TB 350 "ST_ie_3_" {("ie","*-ie+*","ie+*","*-ie").state[3]}
TB 350 "ST_Ie_3_" {("Ie","*-Ie+*","Ie+*","*-Ie").state[3]}
TB 350 "ST_p_3_" {("p","*-p+*","p+*","*-p").state[3]}
TB 350 "ST_P_3_" {("P","*-P+*","P+*","*-P").state[3]}
TB 350 "ST_sil_3_" {("sil","*-sil+*","sil+*","*-sil").state[3]}
TB 350 "ST_Q_3_" {("Q","*-Q+*","Q+*","*-Q").state[3]}
TB 350 "ST_W_3_" {("W","*-W+*","W+*","*-W").state[3]}
TB 350 "ST_i_3_" {("i","*-i+*","i+*","*-i").state[3]}
TB 350 "ST_v_3_" {("v","*-v+*","v+*","*-v").state[3]}
TB 350 "ST_wa_3_" {("wa","*-wa+*","wa+*","*-wa").state[3]}
TB 350 "ST_A_4_" {("A","*-A+*","A+*","*-A").state[4]}
TB 350 "ST_E_4_" {("E","*-E+*","E+*","*-E").state[4]}
TB 350 "ST_S_4_" {("S","*-S+*","S+*","*-S").state[4]}
TB 350 "ST_Z_4_" {("Z","*-Z+*","Z+*","*-Z").state[4]}
TB 350 "ST_l_4_" {("l","*-l+*","l+*","*-l").state[4]}
TB 350 "ST_n_4_" {("n","*-n+*","n+*","*-n").state[4]}
TB 350 "ST_e_4_" {("e","*-e+*","e+*","*-e").state[4]}
TB 350 "ST_t_4_" {("t","*-t+*","t+*","*-t").state[4]}

TB 350 "ST_u_4_" {("u","*-u+*","u+*","*-u").state[4]}
TB 350 "ST_d_4_" {("d","*-d+*","d+*","*-d").state[4]}
TB 350 "ST_O_4_" {("O","*-O+*","O+*","*-O").state[4]}
TB 350 "ST_o_4_" {("o","*-o+*","o+*","*-o").state[4]}
TB 350 "ST_b_4_" {("b","*-b+*","b+*","*-b").state[4]}
TB 350 "ST_a_4_" {("a","*-a+*","a+*","*-a").state[4]}
TB 350 "ST_y_4_" {("y","*-y+*","y+*","*-y").state[4]}
TB 350 "ST_N_4_" {("N","*-N+*","N+*","*-N").state[4]}
TB 350 "ST_m_4_" {("m","*-m+*","m+*","*-m").state[4]}
TB 350 "ST_g_4_" {("g","*-g+*","g+*","*-g").state[4]}
TB 350 "ST_r_4_" {("r","*-r+*","r+*","*-r").state[4]}
TB 350 "ST_w_4_" {("w","*-w+*","w+*","*-w").state[4]}
TB 350 "ST_q_4_" {("q","*-q+*","q+*","*-q").state[4]}
TB 350 "ST_f_4_" {("f","*-f+*","f+*","*-f").state[4]}
TB 350 "ST_ii_4_" {("ii","*-ii+*","ii+*","*-ii").state[4]}
TB 350 "ST_h_4_" {("h","*-h+*","h+*","*-h").state[4]}
TB 350 "ST_hh_4_" {("hh","*-hh+*","hh+*","*-hh").state[4]}
TB 350 "ST_j_4_" {("j","*-j+*","j+*","*-j").state[4]}
TB 350 "ST_k_4_" {("k","*-k+*","k+*","*-k").state[4]}
TB 350 "ST_s_4_" {("s","*-s+*","s+*","*-s").state[4]}
TB 350 "ST_x_4_" {("x","*-x+*","x+*","*-x").state[4]}
TB 350 "ST_C_4_" {("C","*-C+*","C+*","*-C").state[4]}
TB 350 "ST_K_4_" {("K","*-K+*","K+*","*-K").state[4]}
TB 350 "ST_X_4_" {("X","*-X+*","X+*","*-X").state[4]}
TB 350 "ST_Wa_4_" {("Wa","*-Wa+*","Wa+*","*-Wa").state[4]}
TB 350 "ST_z_4_" {("z","*-z+*","z+*","*-z").state[4]}
TB 350 "ST_H_4_" {("H","*-H+*","H+*","*-H").state[4]}
TB 350 "ST_T_4_" {("T","*-T+*","T+*","*-T").state[4]}
TB 350 "ST_ie_4_" {("ie","*-ie+*","ie+*","*-ie").state[4]}
TB 350 "ST_Ie_4_" {("Ie","*-Ie+*","Ie+*","*-Ie").state[4]}
TB 350 "ST_p_4_" {("p","*-p+*","p+*","*-p").state[4]}
TB 350 "ST_P_4_" {("P","*-P+*","P+*","*-P").state[4]}
TB 350 "ST_sil_4_" {("sil","*-sil+*","sil+*","*-sil").state[4]}
TB 350 "ST_Q_4_" {("Q","*-Q+*","Q+*","*-Q").state[4]}
TB 350 "ST_W_4_" {("W","*-W+*","W+*","*-W").state[4]}
TB 350 "ST_i_4_" {("i","*-i+*","i+*","*-i").state[4]}
TB 350 "ST_v_4_" {("v","*-v+*","v+*","*-v").state[4]}
TB 350 "ST_wa_4_" {("wa","*-wa+*","wa+*","*-wa").state[4]}
TR 1

AU

CO

ST