ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

Hidden Markov Model Based Tigrigna Speech Recognition

By:

Hafte Abera Miruts

A THESIS SUBMITTED TO
THE SCHOOL OF GRADUATE STUDIES OF THE ADDIS ABABA UNIVERSITY IN PARTIAL FULFILLMENT FOR THE DEGREE OF MASTERS OF SCIENCE IN INFORMATION SCIENCE

Nov, 2009

ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUSTE STUDIES
FACULITY OF INFORMATICS
DEPARTMENT OF INFORMATION SCIENCE

Hidden Markov Model Based Tigrigna Speech Recognition

By:
Hafte Abera Miruts

APPROVED BY EXAMINING BOARD:

1 Million Meshesha (Ph.D), Advisor _____

2 _____ _____

3 _____ _____

4 _____ _____

5 _____ _____

DEDICATION

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

Acknowledgment

All of my efforts would have gone for naught if it had not been for the importunate help of the Almighty God. I would like to express my sincere gratitude to my advisor Dr. Million Meshesha for his valuable suggestions; constant encouragement, technical support, and patient guidance have been of invaluable importance in the completion of this thesis. I would also like to express my gratitude to Mr. Daniel Yacob, for providing us “transilater package program” that simplifying my work. I would also like to take this opportunity to thank all my friends and my colleagues for their comments on the document and their encouragement during my work.

Finally, I would sincerely like to thank my family for their support and encouragement in my study.

Table of contents

List of Acronyms

AI – Artificial Intelligence

ASR – Automatic Speech Recognition

CSR –Continuous speech recognition

DFT– Discrete Fourier Transform

DTC– Decision tree clustering

EBNF – Extended Backus Naur Format

FFT – Fast Fourier transform

HMM – Hidden Markov Model

HTK – Hidden Markov Model Toolkit

MFCC-Mel-Frequency Cepstral Coefficients

MLF – Master Label File

MMF– Master Macro File

SLF –Standard Lattice Format

SR – Speech Recognition

List of tables

List of Figures

Abstract

This study aims to design speaker independent continuous Tigrigna recognition system. Tigrigna is a very productive language in terms of word forms because of its agglutinative nature. All work is done in HTK (Hidden Markov Model Toolkit) environment, except parsing and network transforming which utilizes perl programming language.

In the process of building the recognizer, the speech data is recorded at a sampling rate of 16KHz and the recorded speech is then converted into Mel Frequency Cepstral Coefficient (MFCC) vectors for further analysis and processing. A corpus has been developed to get the required data for training and testing the models. The corpus is a database comprised of 250 utterances that are used for training and 50 sentences for testing and evaluation. The data is preprocessed in line with the requirements of the HTK toolkit. In order to support the acoustic models, a bigram language model is constructed. In addition, pronunciation dictionary is prepared and used as an input.

Since the recognizer is designed to recognize continuous speech, Phonemes are used as the basic unit of recognition. However phonemes are known to be context independent units, given that the environment in which a sound is put makes a difference in the way it is pronounced. Thus, after the monophone based speech recognizer is built, it is promoted to triphone based system in which the left and right contexts are considered and modeled.

The speech recognizer is then evaluated using the test dataset Performance result shows 60.32% word level correctness, 58.38% word accuracy, and 20 % sentence level correctness are obtained. The results are encouraging and with more optimization works better results can be achieved. To this ends further research works are recommended in line with the analysis and finding of this study.

Key words: *Speech Recognition, HMM, HMM based speech Recognition, Language Modeling.*

CHAPTER ONE

INTRODUCTION

1.1 Background

Designing a machine that mimics human behavior, particularly the capability of speaking naturally and responding properly to spoken language, has intrigued engineers and scientists for centuries. Since the 1930s, when Homer Dudley of Bell Laboratories proposed a system model for speech analysis and synthesis, the problem of automatic speech recognition has been approached progressively, from a simple machine that responds to a small set of sounds to a sophisticated system that responds to fluently spoken natural language and takes into account the varying statistics of the language in which the speech is produced [12].

Research in started in the late 1940s and early 1950s simultaneously in Europe with Dreyfus-Graf who developed the first "Phonetographe" and in U.S.A with K-H Davis who built the first speaker dependent isolated digit recognizer[35]. These systems were electronic devices. Researches in computer-based ASR were started in the late 1950s and early 1960s. The speaker independent vowel recognition system at MIT Lincoln and a system to recognize 10 syllables of a single speaker at RCA laboratories are among the early recognition systems. It was at this time that several Japanese laboratories built special purpose hardware to perform a speech recognition task. In general, the1960s were the time for small vocabulary, isolated word recognition systems developed based on simple acoustic-phonetic properties of speech sound. The 1970s were the time for medium vocabulary recognition system built using simple template-based, pattern recognition methods. The first speech recognition commercial company, which developed VIP-100 system, was also founded during this time. Although VIP-100 has been applied only in a few simple applications, it greatly influenced Advanced Research Projects Agency of the U.S. department of Defense (DARPA) to fund the Speech Understanding Research (SUR) program. Harpy (which was shown to be able to recognize speech using a vocabulary of 1,011 words with reasonable accuracy), Hearsay-

II and HWIM (Hear What I Mean) are among the systems developed under the DARPA's SUR program. It is also at this period that IBM researchers started studying large vocabulary speech recognition for different tasks, notably for voice activated type writer, and AT&T Bell laboratory researchers started to study a speaker independent speech recognition to provide automated telecommunication service [35].

In 1980s researchers started to tackle large vocabulary speaker-independent, continuous speech recognition problems based on statistical methods using technologies such as hidden Markov models (HMM). Speaker-independence has been possible because of HMM's ability of capturing and modeling speech variability. Artificial neural networks (ANN) were also reintroduced in the late 1980's. Recording of large speech databases such as TIMIT, which contributed to the advances made in ASR, was done in this period. Based on major advances in statistical modeling of speech in the 1980s, automatic speech recognition systems today find widespread application in tasks that require a human-machine interface, such as automatic call processing in the telephone network and query-based information systems that do things like provide updated travel information, stock price quotations, weather reports, etc. In 1990s it was possible to build large vocabulary systems with unconstrained language models, and constrained task syntax for continuous speech recognition and understanding [11].

The success of statistical methods revived the interest from DARPA leading to several speech recognition systems such as the Sphinx system, the BYBLOS system and the DECIPHER system from CMU, BBN and SRI respectively. As a result, researchers developed an increasing interest for speech processing under noisy or adverse conditions, spontaneous speech recognition and dialog management [12].

In the first half of 2000s, the attention of researchers has been attracted towards spontaneous, robust, multimodal speech recognition, although the accuracy drastically decreases for spontaneous speech. On the other hand, speech recognition systems applied for real applications should have high accuracy for spontaneous speech and should also be robust.

Human beings use multimodal communication when they speak with each other especially in noisy environments. Speech recognition systems developed using audio and visual information showed better recognition performance than systems developed using only speech information. During this period, DARPA conducted The Effective Affordable Reusable Speech-to-Text (EARS) program so as to develop automatic transcription with the aim of achieving more accurate output [2].

As it can be observed from the above explanation, speech recognition technology has gone through noteworthy progress. The use of HMM, the development of large speech corpora for system development training and testing, establishment of standards for performance evaluation, and advances in computer technology are the factors that fueled the progress.

Although lots of work has been conducted in the area of ASR for technologically favored languages for more than 50 years, research in local language speech recognition started only recently, 10 years ago. Any speech recognition system is the building of a speaker independent system that recognizes in real-time, natural, fluent, spontaneous, continuous speech and which is capable of telling speech and background noise apart. However, the current generation of systems, even in the developed languages, is nowhere near this ideal [25].

1.1.1 Automatic Speech Recognition over view (ASR)

Speech is the primary means of communication between people and it is how human should be able to interact with computers. For reasons ranging from technological curiosity about the mechanisms for mechanical realization of human speech capabilities, to the desire to automate simple tasks inherently requiring human-machine interactions. Speech interface in user's own language is in short, an ideal means of communication as being the most natural, flexible, efficient and convenient option allowing hands and eyes to be free [22].

With in the area of human – machine communication by voice, there are three main avenues of research [25]; namely,

- Speech synthesis (Voice output);
- Speaker recognition (voice recognition); and
- Speech recognition (voice input)

Speech synthesis is a technology for generating voice output from an electronic text. It converts text to speech and read for the user. Speaker recognition, on the other hand, involves identifying a specific speaker out of a known population, or verifying the claimed identity of a user, thus enabling controlled access to locales and services [45].

Computer based speech recognition is defined as the ability to take speech as input and produce a transcript of that speech as output. In technical terms, it is the conversion of an acoustic signal to a stream of text words [28].In addition Automatic Speech Recognition (ASR) is a process by which a computer identifies spoken words from an input voice device. Basically, SR means talking to your computer, and having it correctly recognize what you are saying [4].

1.1.2 Speech Recognition systems

Speech recognition systems can be separated in several different classes by describing what types of utterances they have the ability to recognize [41]. These classes are based on the fact that one of the difficulties of ASR is the ability to determine when a speaker starts and finishes an utterance. Most packages can fit into more than one class, depending on which mode they are using.

A. Isolated Words Vs Connected Words Vs Continuous Speech system

Isolated word recognizers usually require each utterance to have quiet on both sides of the sample window or lack of an audio signal [1]. It does not mean that it accepts single words, but does require a single utterance at a time, where they require the speaker to wait between utterances. on the other hand connect word systems are similar to isolated

words but allow separate utterances to be run together with a minimal pause between them[17]. Recognizers with continuous speech capabilities are some of the most difficult to create because they must utilize special methods to determine utterance boundaries. Continuous speech recognizers allow users to speak almost naturally, while the computer determines the content. Basically, it is like computer dictation.

B. Large Vocabulary Systems vs. Small Vocabulary Systems

Vocabularies are lists of words or utterances that can be recognized by the speech recognition system [50]. There are no established definitions, however small vocabulary has tens of words, medium vocabulary has hundreds of words, large vocabulary has thousands of words and very-large vocabulary has tens of thousands of words.

Generally, smaller vocabularies are easier for a computer to recognize, while large vocabularies are more difficult [43]. As the vocabulary size gets larger, the number of confusable words grows substantially and also affects the complexity, processing requirements and the accuracy of the system. However, with small vocabularies, each word can be modeled individually, because it is reasonable to expect sufficient training for the handful of words.

C. Speaker dependent system Vs Speaker independent systems

A speaker dependent system is built up to operate for speaker/speakers that are involved in the training. To learn the model parameters of the speakers' voice through training it uses speech from the target speaker [26]. Speaker dependent systems are usually easier to develop, cheaper to buy and more accurate [26]. Speaker dependent systems are useful for some applications such as dictation system [16]. However speaker independent systems are capable of recognizing speech from any new speaker. It is used by any user, with no enrollment or prior training. Such systems in general are said to be most difficult to develop because those applications must recognize large, heterogeneous populations of speakers [18]. However, they are more flexible and natural.

1.1.3 Different Approaches to Speech Recognition

Speech recognition systems assume that the speech signal is a realization of some message encoded as a sequence of one or more symbols. The basic goal is to" decode" this message and then convert it either into writing (e.g. dictation machine)or into commands to be processed (e.g. hands free dialing).

There are three approaches to speech recognition;

1. The acoustic-phonetic approach
2. The pattern recognition approach
3. The stochastic approaches

The acoustic-phonetic approach is based on the theory of acoustic phonetics. Acoustic phoneticians study our speech by transforming it into an electrical signal with the adequate transducer: a microphone. In modern recording systems, the resulting electrical signal is then digitized: it is successively low-pass filtered, sampled, and quantized. It may then be submitted to various digital signal-processing operations, so as to highlight its acoustic traits: fundamental frequency (often denoted as F0), intensity, and spectral energy distribution. Each acoustic trait is itself related to a perceptual quantity: pitch, loudness, and timbre [23].

The theory proposes that there exist finite, distinct phonetic units in spoken language. The phonetic units are characterized by a set of properties that are embedded in the speech signal or its spectrum over time. Acoustic-phonetic is the oldest, most straightforward and thoroughly researched method, whose principles are still used in the AI based recognizers .The approach assumes that the rules governing the phoneme variability are relatively simple and easily learnable. Even though the acoustic properties of phonetic units are highly variable, both with speakers and with neighboring phonetic units so called co-articulation of sounds, it is assumed that the rules governing the variability are straightforward and can readily be learned and applied in practical situation [15].This approach has restrictions such as absence of well-defined automatic procedure and standard linguistic ways that label the training speech. Due to these, the

acoustic phonetic based speech recognition is no longer considered as the most interesting approach [27].

In pattern-recognition approach to ASR, the speech patterns are used directly as features for representation of speech segment. The method has two steps [24]: Training of speech patterns and recognition of patterns via pattern comparison. Speech knowledge is supplied into the system via the training procedure. Simple pattern matching techniques are not suitable for continuous speech recognition because segment boundaries are difficult to detect and this approach is best suited to small vocabulary speaker dependent recognition [24].Current ASR systems are based on the principles of statistical pattern recognition. The basic methods of applying these principles to the problem of speech recognition were proposed by Baker and Jelinek in 1970's [24].

The stochastic approach entails the use of probabilistic models to deal with uncertain and incomplete information found in speech recognition. The term stochastic refers to the process of making a sequence of non-deterministic selections from among sets of alternatives. They are non-deterministic because the choices during the recognition process are governed by the characteristics of the input and not specified in advance. Both template matching and this approach require the creation and storage of models of each item to be recognized. However, stochastic processing involves no direct matching between stored models and input. Instead it is based upon complex statistical and probabilistic analyses that are best understood by examining the network like structure in which those statistics are stored. HMM, one type of stochastic process model is the basis of the majority of current speech recognition systems [24].

However, the artificial intelligence approach is a compound approach that utilizes the ideas of the first two approaches. The intention here is to mechanize the recognition procedure like the way a person applies his intelligence in analyzing and making decision on the acoustic knowledge. The aim is to integrate phonemic, lexical, semantic and pragmatic knowledge together [15]. Most important techniques in the AI approach are the use of an expert system for segmentation and labeling of the acoustic signal; learning and

adaptation over time; and the use of artificial neural networks (ANNs) or distinguishing between similar sound classes and learning the relations between all known inputs and phonetic data. The use of neural networks is regarded as a separate structural approach that can be used in each of the above approaches. This leads to a hybrid structure of ASR systems [15].

1.2 Statement of the Problem and Justification

The ultimate goal of any research in speech recognition is to develop a system that can understand the speech of anyone who needs to use the system. To meet this requirement, the system must be able to handle inter-speaker and intra-speaker variability, recognize unrestricted vocabulary and understand fluent, conversational speech [16]. Although research in speech technology is about five decades old, such a system has only been a dream so far. However, intensive researches have been going on toward that end. It is certain that human beings from the early days on have gone through various social and technological developments [36].

Speech recognition is a rich field for both practical and intellectual reasons. As a practical area, it solves problems, improve productivity, gaining rapid return on investment, access to new markets which helps render 24-hours service ,environment control like for disabled people or eyes and hands busy working environment and change the way we run our lives. Intellectually, it holds considerable promise as well as challenges in the years to come for scientists and product developers alike. From the practical point of view, speech interfacing serves a lot more problems of human beings, speedup the way people lead their day to day life and in turn help maximize productivity. Above all, speech is the most natural and simple way of communication mechanism ever. One can imagine how it feels like to have a friendly machine that lends wide-open ear, patient, amusing and serving [27].

The development of machines/technologies to the services of human in particular, has been growing so fast. Apart from the design and use of the serving machines, the make an effort to domestic these machines and let them act like human is incredibly bearing fruit.

One aspect of such attempts is letting machines recognize human speech in such away that they will be able to understand and respond to a human friend. Such endeavors and attempts are never without reasons. Speech is the most natural means of communication between humans. One of the first skills we learn to use from our babyhood and which can be done without any tools or any explicit education. It is clear that before the invention of writing, speech was the only way of passing knowledge. Even in the modern world, despite all our novel ways of communicating, the speak-listen style is chosen most. It is also the most important way of communicating. When speaking with somebody, one does not have to focus on the audience; he/she can look in a different direction or even perform some other task while communicating. So it is only logical that machine interface designers in their quest for a natural man-machine interface have turned to automatic speech recognition and speech production as one of the most promising interfaces [34].

Speech recognition systems that have been developed so far are meant to serve a specific language such as English, French, and Chinese and are totally limited to some techno-rich countries of the world [21]. For developing countries like Ethiopia, it is only mandatory to follow suit of those developed nations in relation to such technological advancements; or else they will be bound to stay at the bottom of progress because they fail to exploit the opportunities offered by technologies.

So far, little has been done in relation to speech technology on all local languages. It is only in the past ten years that glimmers of such works have started to sparkle for Amharic. The works by Solomon [34], Zegaye [45] and Kinfe [14] are the only mentionable deeds and that actually light a candle in the area .Solomon [34] has tried to indicate the possibility of isolated Amharic consonant-vowel (CV) syllable recognition in his work. Kinfe [14] has moved a step forward; he examined and compared the potential of three basic units of recognition phoneme, tri-phone and CV syllable .He did also indicate the possibility of discrete sub-word based Amharic word recognition. Zegay [45] has tried on Hidden Markov Model Based Large Vocabulary, Speaker independent, and Continuous Amharic speech recognition: he examined and compared

the difference between speaker dependent and Speaker independent, Small or Large Vocabulary and Discrete and continuous speech recognition [45]. On the other hand to the best knowledge of the researcher studies in Tigrigna speech recognition are none.

According to the Office Population and Housing Census Commission of Ethiopia (2007) there are about 4.5 million speaker of Tigrigna. They make up approximately 96.6% of the inhabitants of the Tigray Region , and are 6.2% of the population of Ethiopia as a whole, In addition Tigrinya speakers are approximately 50% of the population in neighboring Eritrea at about 2.25 million people. So that, the total number of Tigrigna speaker in Tigray and Eritrea was 6.75 million. At the present Tigrigna is the second most widely spoken Semitic language next to Amharic in Ethiopia.

Tigrigna is an agglutinative language and is highly productive in terms of word generation. For instance, from a verbal stem one can generate hundreds of distinct words by concatenating various suffixes, each of them including different meanings. In addition, syntax rules in Tigrigna are not well defined, i.e. that is why it is hard and not straightforward to implement a Tigrigna speaker independent continuous system based on the studies made for languages like English. Taking into account the term “continuous”, there is not a system developed for Tigrigna, which processes naturally spoken utterances. This thesis therefore attempts to build a speaker independent continuous Tigrigna speech recognizer using HMM that can enable to exploit the potential of ASR for local language.

Research Questions

To find solution for the above problem, this research attempts to provide solution to the following question.

1. How a system that recognizes multiple speakers can be built?
2. What is the challenge if the speech is entered as a continuous utterance rather than discrete units?

3. Is the system to be operated in a quiet or noisy environment, and what is the nature of the environmental noise if it exists?
4. What are the linguistic constraints to be placed upon the speech, and what linguistic knowledge to be built into the recognizer?

1.3 Objectives of the Study

1.3.1. General Objective

The general objective of this research is to investigate and demonstrate the possibility of developing speaker-independent continuous Tigrigna speech recognizer using Hidden Markov Model.

1.3.2. Specific Objectives

To accomplish the above general objective, the following specific objectives have been targeted.

- To understand conceptual and design issues in speech recognition system in general and on how to develop a recognizer for Tigrigna Language using Hidden Markov Model Toolkit in particular by reviewing literature and documents or manuals.
- To examine features of the spoken Tigrigna speech in view of realizing automatic speech recognizer.
- To investigate the various speech recognition techniques such as labeling, signal analysis and feature extraction to develop the recognizer.
- To build a prototype speaker-independent continuous Tigrigna speech recognizer using Hidden Markov Model (HMM).
- To evaluate the performance of the prototype recognizer and report the findings.
- To provide concluding remarks and forward recommendations for further study.

1.4. Methodology

In order to achieve the objective stated in section 1.3, this study employ the following methods for conceptual understating, data set preparation, designing the system and evaluation of the prototype speech recognizers.

1.4.1. Review of Related Literature

A number of resources such as books, research reports, journal, articles and other published and unpublished documents have been used so as to:

- Understand speech, speech production.
- Understand phonology and linguistics of Tigrigna language, especially those dealing with the phonetic/tri-phone features is reviewed.
- Examine and select types of speech, and acoustic units to be modeled.
- Compare and contrast speech signal representations.
- Identify and study statistical speech recognition models.
- Identify appropriate tools required to develop a prototype.

Moreover, to learn what others have done in the area and to better understand the problem, a comprehensive exploration of available empirical literature on automatic speech recognition is conducted.

1.4.2. Data selection and Preparation Methods

In order to build a robust speaker independent continuous speech recognizer, it is crucial that all acoustic (sub-word) models receive enough training examples, taking into account the many variations that can occur in speech. It follows that a training corpus should be sufficiently large, consisting of speech samples spoken by men and women from different age. Since the sound of a sub-word unit is also influenced by its context the speech data should include all phonemes in as many different phonetic contexts as possible. In practice this means that, depending on the quality of the data and the complexity of the acoustic models, phonetically rich sentences are necessary [41].

Recording such dataset obviously is a major undertaking involving sub-tasks like selection of phonetically rich and phonetically balanced sentences, selection of appropriate participants, recording the data and the most time-consuming parts post-processing and transcribing the data. Studies of this kind in the developed languages are fortunate enough that many of these speech corpora are commercially available. However no such commercial database is there for Tigrigna. So that the researcher has employed random sampling and has capture a corpus from 12 people with the required variety and each speaker having read 25 sentences, summing up to a total of around 300 sentences. The data set composion shown in the table1.1.

	Sex	Age
Male	8	20-60
Female	4	25-45

Table 1.1 Composion of data set

1.4.3. Designing Technique

The recognizer is built using the popular Hidden Markov Model (HMM).HMM is a powerful technique capable of robust modeling of speech. It is a parametric model that is particularly suitable for describing speech events. HMMs are also a succinct representation of speech events; therefore, they require less storage than many other strategies. Markowitz [18] further noted that, HMM is the most widely applied and most successful speech modeling technique. Solomon [34] and Kinfe [14] also applied HMMs for designing speech recognizer for Amharic language.

HMMs are used in speech recognition in that a speech signal could be viewed as a piecewise stationary signal or a short-time stationary signal. That is, one could assume in a short-time in the range of 10 milliseconds [41]. In speech recognition, the Hidden Markov Model would output a sequence of *n*-dimensional real-valued vectors (with *n* being a small integer, such as 10), outputting one of these every 10 milliseconds. Each phoneme model in this research is HMM. The appropriate tools from the HTK toolkit are

used for the development purpose. The tools in HTK Provide sophisticated facilities for speech analysis, HMM training, testing and evaluation.

1.4.4. Evaluation and Testing Techniques

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

1.5 Applications of the Study

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

- People with disabilities are another part of the population that benefit from using speech recognition programs. It is especially useful for people who have difficulty with or are unable to use their hands, from mild repetitive stress injuries to involved disabilities that require alternative input for support with accessing the computer.
- Computer Aided Instruction systems do make use of ASRs. For developing countries in particular, where majority of the people are illiterate, ASR can help solve many problems like on education, communication, storage and retrieval of orally available knowledge .
- Dictation systems like automatic typewriter that is activated and work by voice.

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

1.6 Scope and Limitations of the Study

The primarily focus of the researcher is to investigate the possibility of developing a prototype speaker independent and continuous speech recognizer for Tigrigna language.

This study has attempted to investigate the basic units of recognition for Tigrigna and has identified possible challengers. Prototype Tigrigna phonemes recognizers is built for both speaker-dependent and speaker-independent models using these contending sub-word units. Last but not least, comparative analysis of these contending sub-word units has been made based on the recognition performance of the respective recognizers. Two of the identified sub-word units proved to be strong contenders and have been subjected to further examination and conclusions have been drawn based on the results obtained from the examination.

The scope of the study is limited to HMM base speaker independent continuous Tigrigna phonemes recognition and it is based on pure acoustic phonetic information and the term recognition is used as distinct from understanding where no linguistic and semantic analysis is involved.

The main limitation of this study was the absence of a ready-made Tigrigna reference database or corpus for speech recognition research. This paper had to go to the extent of preparing speech database and deal with the rest of the experiment based on this small database. Though every possible effort has been made to impart good coverage and phonetic balance into the 'database', experimental results have demonstrated that more

effort is required to prepare a phonetically balanced database with good coverage to come up with an efficient speech recognition system.

The other constraint associated with building the database has been human and environmental factors had their own impact on the results of the experiments because some of the speakers are stressed and are not consistent in their utterances and the recording laboratory is noisy and is not favorable for the research. The last and the obvious limitation is time which constrained the researcher to select 300 sentences.

1.7 Organization of the Thesis

This paper is organized in six chapters. Chapter one introduces the basic concepts of speech recognition including definition of terms and application areas of speech recognition. Furthermore, statements of the problem, justification of the study, objectives of the research, and methods employed for the successful completion of the study are briefly presented.

Chapter two presents the phonology of Tigrigna language, Tigrigna writing system and recognition units have been introduced very briefly.

Chapter three covers literature review on speech recognition system, signal processing, Acoustic model and representation of speech are briefly introduced. Furthermore, the Dynamic Time Warping, Hidden Markov Models, language model and related works are briefly discussed in this chapter.

Chapter four deals with techniques for designing Tigrigna speech recognizer and introduces the HTK toolkit briefly.

Chapter five discusses the design and implementation details of the Tigrigna phonetics recognizers using different sub-word units as the basic units of recognition and the results obtained are discussed.

Chapter six presents the conclusions drawn out of the experiments and forwards recommendations for future work.

CHAPTER TWO

PHONOLOGY OF TIGRIGNA AND THE WRITING SYSTEM

2.1. Introduction

This chapter presents the Tigrigna language phonetics and writing system. Language is one of the tools for humans to pass their message to others. Each language has got its own set of phonemes from which sounds of words are generated. Using set of words, we can form a phrase; and a combination of phrases form a sentence. Hence, it is possible to say that the basic unit for a language is the phoneme set and due attention should be given to phonetics. Phonetics is the study of the physical sounds of human speech and it is concerned with the physical properties of phonemes, and the processes of their physiological production [47].The phonetic alphabet is usually divided in two main categories, vowels and consonants. Vowels are always voiced sounds and they are produced with vocal cords in vibration, while consonants may be either voiced or unvoiced. Vowels have considerably higher amplitude than consonants and they are rapid changes [32].

2.2. The Phonology of Tigrigna

The language Tigrinya (ትግርኛ, tigriññā), also spelled Tigrigna orTigrinyan, is one of the North-Ethio-Semitic Languages which include Geez,Tigre and Tigrigna spoken by the Tigray-Tigrinya people in Tigray Northern Ethiopia it also widely spoken in central Eritrea , where it is one of the official languages of Eritrea[6].

Tigrigna language has its own characterizing phonetic, phonological and morphological properties. It has a set of speech sounds that is not found in other languages. For example the following sounds are not found in English and Amharic: [ʔ](ዐ),[ḥ] (ሐ),[k'](ኸ),[ʔ](ኣ) and [x'] (ቐ).Tigrigna also has its own inventory of speech sounds. For example the classification of Tigrigna consonants and vowels Languages differ in their set of phonemes [6].

2.2.1 Consonant Phonemes

Tigrinya has a fairly typical set of phonemes for an Ethiopian Semitic language. That is, there is a set of ejective consonants and the usual seven-vowel system. Consonants are created when the airflow is directly restricted, or obstructed, so that air can not escape without creating friction that can be heard [29].

There are twenty-nine consonant phonemes in Tigrigna .The consonants are generally classified as Stops, fricatives, nasals, liquids, and semi-vowels [6]. Unlike many of the modern Ethiopian Semitic languages, Tigrigna has preserved the two pharyngeal consonants which is apparently part of the ancient Ge'ez language and which, along with [x'], a velar or uvular ejective fricative, make it easy to distinguish spoken Tigrinya from related languages such as Amharic. The Appendix A shows the phonemes of Tigrinya. The consonant /v/ appears in parentheses because it occurs only in recent borrowings from European languages. The fricative sounds [x], [x^w], [x'] and [$x^{w'}$] occur as allophones [6].

All consonants can be germinated except the pharyngeal and laryngeals. The term geminate consonant refers to, a linguistic term meaning the doubling of a consonantal sound, is meaningful in Tigrinya, i.e. it affects the meaning of words ,for example the ‘d’ in the word giddaf (ግደፍ) is a geminate consonant [6]. While gemination plays an important role in the morphology of the Tigrinya verb, it is normally accompanied by other marks. But there is a small number of pairs of words, which are only differentiable from each other by gemination, e.g. k'ärräbä, "he presented, he brought forth"; k'äräbä, "he came closer". All the consonants, with the exception of the pharyngeal and glottal, are amenable to gemination. Consonants are classified according to Manner of Voicing, places of articulation, manners of articulation [25].

2.2.1.1 Manner of Voicing

Refers to whether the vocal cards vibrate or not. The level of vibration of the vocal cords determines whether a sound is voiced or unvoiced.

Unvoiced: If the vocal cords are apart, then air can escape unrestricted. Sounds produced

Voiced: if the vocal cords very close together, the airs blow them apart as it forces its way through. This makes the cords vibrate, producing a voiced sound. The sound[S(ሰ)] is called voiced because the vocal cords do vibrate.

2.2.1.2 Place of Articulation

The air stream used in producing speech sounds comes from the lungs and passes through the larynx and the vocal cords. Some of the components of the sound so produced are filtered out by them [25].Tigrigna consonants may be classified by the place of articulation as bilabial (lip together) labiodentals (lower lip against front teeth), dentals (teeth together), palatals (tongue on hard palate) velars (tongue near velum) and glottal (space between vocal folds).

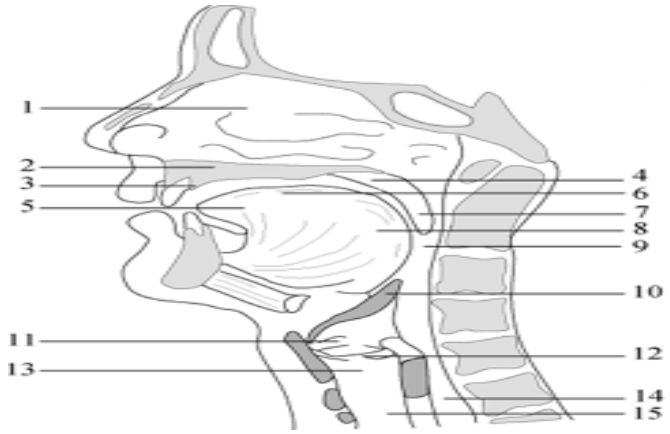


Figure 2.1: human vocal organs adapted from [31]

The label for each vocal organs: (1) Nasal cavity, (2) Hard palate, (3) Alveoral ridge, (4) Soft palate (Velum), (5) Tip of the tongue (Apex), (6) Dorsum, (7) Uvula, (8) Radix, (9) Pharynx, (10) glottis, (11) False vocal cords, (12) Vocal cord, (13) Larynx, (14) Esophagus, and (15) Trachea.

Human speech is produced by vocal organs presented in Figure 2.1. The main energy source is the lungs together with the diaphragm. When we speak, the air flow is forced through the glottis (10) between the vocal cords (12) and the larynx to the three main

cavities of the vocal tract: the pharynx (9), the oral and nasal cavities (1). From the oral and nasal cavities, the air flow exits through the nose and mouth, respectively. The V-shaped opening between the vocal cords, called the glottis (10), is the most important sound source in the vocal system. The vocal cords may act in several different ways during speech. It is used to modulate the air flow by rapidly opening and closing so as to cause buzzing sound from which vowels and voiced consonants are produced [33]. The pharynx (9) connects the larynx to the oral cavity. It has almost fixed dimensions, but its length may be changed slightly by raising or lowering the larynx at one end and the soft palate (4) at the other end. The soft palate also isolates or connects the route from the nasal cavity to the pharynx. At the bottom of the pharynx are the epiglottis and false vocal cords (11) to prevent food reaching the larynx and to isolate the esophagus acoustically from the vocal tract. The epiglottis (10), the false vocal cords and the vocal cords are closed during swallowing and open during normal breathing. The oral cavity is one of the most important parts of the vocal tract. Its size, shape and acoustics can be varied by the movements of the palate, the tongue, the lips, the cheeks and the teeth. Especially the tongue is very flexible, the tip and the edges can be moved independently and the entire tongue can move forward, backward, up and down. The lips control the size and shape of the mouth opening through which speech sound is radiated. Unlike the oral cavity, the nasal cavity has fixed dimensions and shape. The air stream to the nasal cavity is controlled by the soft palate [32].

2.2.1.3 Manner of Articulation

The source of speech sounds is the air passing from the lungs through the and into the cavity in the oral tract. These cavities can be modified in shape by the different position of the articulators (the tongue, lips, velum etc).the way the shape of the cavity changes influences the way the air in it vibrates giving rise to different sounds [25].Tigrigna consonant may be classified by the manner of articulation as Stops, Fricatives , Affricatives ,Nasals ,Liquids ,and Semi-Vowels.

- **The Stops:** are produce by a complete closure blocking the air commentaries and then releasing it abruptly [25].Stops are also plosives. There are ten stop

phonemes out of which three[b(በ),d(ደ),g(ገ)] are voiced and the rest [p(ፐ),t(ተ),k(ከ),p(ጰ), q(ቀ),ʔ(ዐ)] are unvoiced phonemes [p(ፐ),T(ተ),q(ቀ)] are the glottalized counterparts of [p(ፐ),t(ተ),k(ከ)].

- **The Fricatives**: is made by an incomplete closure (or stricture) which produces friction as the air is forced through its [25].Fricatives are also termed continuants. there are nine fricative phonemes out of which three[z(ዘ),ž(ዠ),'(ኣ)] are voiced and the other six [f(ፈ),S(ሸ),'p'(ጰ),š (ሠ),h(ሀ),ḥ (ሐ)] are unvoiced. In addition the velar consonants /k/ and /k'/ are pronounced differently when they appear immediately after a vowel and are not geminated. In these circumstances, /k/ is pronounced as a velar fricative. /k'/ is pronounced as a fricative, or sometimes as an affricate. This fricative or affricate is more often pronounced further back, in the uvular place of articulation, it is represented with [x'].

All of these possible realizations - velar ejective fricative, uvular ejective fricative, velar ejective affricate and uvular ejective affricate - are cross-linguistically very rare sounds. Since these two sounds are completely conditioned by their environments, they can be considered allophones of /k/ and /k'/. This is especially clear from verb roots in which one consonant is realized as one or the other allophone depending on what precedes it. For example, for the verb meaning cry, which has the tri-consonantal root {bky}, there are forms such as ምብኻይ məbkay (to cry) and በኸየ bäxäyä (he cried), and for the verb meaning steal, which has the triconsonantal root {srk'}, there are forms such as ይሰርቁ yəsärk'u (they steal) and ይሰርቕ yəsärrəx' (he steals).What is especially interesting about these pairs of phones is that they are distinguished in Tigrinya orthography. Because allophones are completely predictable, it is quite unusual for them to be represented with distinct symbols in the written form of a language [48].

- **The Affricates**: combines a stop with a following continuant but lasts only as long as a single fricative [25].There are three affricate phonemes [ǧ(ጀ),č (ቸ),č' (ጨ)] which are voiced, unvoiced and ejective respectively. The(C) is the glottalized counter part of (c).

- **The Nasals:** is formed when air escapes through the nose. For this to happen, the soft palate is lower to allow air to pass it, whilst a closure is made in the oral cavity to stop air escaping through the mouth [25] .there are three nasal phonemes [m(ም),n(ነ).ñ (ኘ)].

- **The Liquids**: is formed by partial blockage or obstruction of air in the mouth. Liquids consist of lateral [l(ለ)]sounds and flap [r(ረ)]sounds[25].both are voiced and dental phonemes and they occurs in all positions.

- **The semi-vowels**: are vowels like consonants. they do not involve a blockage or obstruction of air in the mouth as is the case with stops and fricatives[25].there are two semi-vowels[w(ወ),y(የ)] in Tigrigna and they occur in all positions.

All the twenty-nine consonant phonemes of Tigrigna, which have been discussed above, are presented here in the tabular form [6].

Consonants								
		Bilabial / Labiodental	Dental	Palato-alveolar/ Palatal	Velar		Pharyngeal	Glottal
					Plain	Labialized		
Stops and affricates	Voiceless	p(ፐ)	t (ተ)	č [̆] (ቸ)	k(ከ)	Kʷ(ኰ)		ʾ [ʁ](አ)
	Voiced	b (በ)	d (ደ)	ǧ [̂] (ጀ)	g(ገ)	Gʷ (ጎ)		
	Ejective	p' (ጰ)	t' (ጠ)	č' [̆'] (ፚ)	k'(ቀ)	Kʷ'(ቈ)		
Fricatives	Voiceless	f (ፈ)	s (ሰ)	š [ṯ] (ሸ)	(x) (ኸ)	(xʷ)(ኼ)	ḥ [ħ] (ሐ)	h(ሀ)
	Voiced	(v) (ቨ)	z(ዘ)	ž [ʒ] (ዠ)			ʿ [ɘ] (ዐ)	
	Ejective		s'(ፀ)		(x') (ቐ)	(xʷ') (ቈ)		
Nasals		m (ም)	n(ነ)	ñ [ɲ] (ኘ)				
Approximants			l(ለ)	y [j] (የ)		W(ወ)		
Flap/Trill			r(ረ)					

Table 2.1 Consonants with their feature

2.2.2 Vowel Phonemes

One of the basic differences between consonants and vowels is that in the case of vowels, the air generated by lung will vibrate the vocal cord since the vocal cord is slightly closed when vowels are generated. Moreover, in generating different vowels the tongue plays an important role. The contribution of tongue can be seen in two different aspects [6]. The first one is the way the tongue is positioned in terms of height and the second one is the movement of tongue to the front and back of the mouth. In addition to tongue, the shape of lips has also a contribution in changing the vowels sound when it varies from rounded to non-rounded (flat).In addition Vowels differ from consonants in that there is no noticeable obstruction in the vocal tract during their production. Air escapes in a relative unrestricted way through the mouth and/or nose [25]

In Tigrigna there are seven vowels. These are **ኧ, ኡ**, **ኢ, ኣ** , **ኤ**, **እ**, and **ኦ**. All are voiced and oral sounds. These vowels can be found in each letters, that is, each letter in Tigrigna is not a single sound rather they are a combination of two sounds, one from vowel and one from consonant [6].

These vowels can be divided into different categories depending on how they are formulated; Front, Central or Back position of the tongue, wideness /roundness of the constriction position, and place of the tongue (high, mid or low) [29]. Tigrigna vowels and their categorization are summarized in table 2.2.

The Front Vowel: these are two in number [i(ኢ),e(ኤ)] both are un rounded .The mid front vowel(e)does not occur word finally while the high front(i) occurs medially as well as finally.

The central Vowels: there are three central vowel [(ə) [ɨ](ኧ),**a**(ኣ),a(ኣ)] which are all un rounded .Among these (i) never occurs word finally, where as the remaining two are found medially and finally.

The Back vowels: these are [u(ኡ),o(ኦ)].Both are rounded and they occur in word medially and word final positions[6].

Vowels			
	Front	**Central**	**Back**
High	i(ኢ)	ə [ɭ] (እ)	u(ኡ)
Mid	e(ኤ)	ä [ɛ] [ኧ]	o(ኦ)
Low		a(አ)	

Table 2.2 vowels with their feature

2.3 The Tigrigna Writing System

Tigrigna writing system is phonetic,[48] meaning, anyone who wants to write Tigrigna text can write the text as long as he/she can speak the language and has a good working knowledge of Ethiopic script. Unlike most languages, no one needs to learn how to spell Tigrigna words, nor to see a word first written in order to know how to spell it.Tigrinya is written using the Ge'ez script, originally developed for the now-extinct Ge'ez language. Tigrigna script (orthographic representation) consists of 40 core characters as shown in Appendix A. Each consists of seven vowels of Tigrinya (and Ge'ez); they appear in the traditional order. The rows are assigned to the consonants, again in the traditional order. For each consonant in an abugida, there is an unmarked symbol representing that consonant followed by a canonical or inherent vowel. For the Ge'ez abugida, this canonical vowel is /ä/, the first column in the table. This representation of each character in 7 different forms makes the number of core characters to be 244, also called syllographs (ፊደል) including twenty numbers and eight punctuation marks will be 272 [5] However, since the pharyngeal and glottal consonants of Tigrinya (and other Ethiopian Semitic languages) cannot be followed by this vowel, the symbols in the first column in the rows for those consonants are pronounced with the vowel /a/, exactly as in the fourth row. These redundant symbols are falling into disuse in Tigrinya and are shown with a dark gray background in Appendix A. When it is necessary to represent a consonant with no following vowel, the consonant+ə form is used (the symbol in the sixth column). For example, the word ’əntay 'what?' is written እንታይ,literally ’ə-nə-ta-yə. Since some of the distinctions that were apparently made in Ge'ez have been lost in Tigrinya, there are two rows of symbols each for the consonants /h/, /s/, and /s'/. In Tigray, for /s/ and /s'/, at

least, one of these has fallen into disuse in Tigrinya and is now considered old-fashioned. These less-used series are shown with a dark gray background in the chart. The orthography does not mark gemination, so the pair of words k'ärräbä 'he approached', k'äräbä 'he was near' is both written ቀረበ. Since such pairs are very rare happen, it has no that much significant input to readers of the language [48].

The entire table where these consonants and their orders are listed is known as FIDEL. Two of the Tigrigna consonants with their seven forms and orthographic symbols are given below:

Orders:	1st	2nd	3rd	4th	5th	6th	7th
Transcription:	bä	bu	bi	ba	be	bə	bo
Orthographic symbol:	በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ
CV representation :	ብአ	ብኡ	ብኢ	ብኣ	ብኤ	ብእ	ብኦ
Transcription:	dä	du	di	da	de	də	do
Orthographic symbol:	ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ
CV representation:	ድ አ	ድ ኡ	ድ ኢ	ድኣ	ድ ኤ	ድ እ	ድኦ

2.4 Recognition Units

Speech recognition systems assume that a speech signal is an understanding of some message encoded as a sequence of one or more symbols [42]. The role of a recognizer in a recognition system is, therefore, to identify the underlying symbol sequence for a given utterance [18]. In order to design a speech recognizer for any language, one must first establish some way of symbolizing the spoken utterances by some group of symbols that distinctly represent the sounds produced. Words are the most natural units of speech because they are exactly what we want to recognize. Moreover, word models are able to capture within-word contextual effects [18]. Therefore, for small-vocabulary systems word models will usually yield the best performance. However, using word models in large-vocabulary recognition introduces several problems. One of these problems is that since training data cannot be shared between words, each word has to be trained individually and each word has to be uttered several times consistently. Yet, this is

impractical since there are too many words to train adequately. In addition to this, the same word spoken with different pitches (high or low) may be recognized as different words and may degrade performance.

These problems can be alleviated, at least in theory, if smaller units, called sub-word units, are used to form the basis of the recognition process. Taking this fact into consideration, and since the intention of the project is to build a large vocabulary system, the first task undertaken is to identify good sub-word units that can be used to build the recognizer. Additionally, to obtain reliable whole word models, the number of word utterances in the training set needs to be sufficiently large. This means that each word in the vocabulary should appear in every possible context several times in the training set. Collecting such an amount of training data and training the models is not practical.

Another problem with using whole-word models in a speaker independent continuous speech recognition system is that the phonetic content of the individual words will overlap. So, storing and comparing whole word patterns will be redundant. The sub-word units are employed in state of the art systems.

Good sub-word units should be consistent and trainable [17]. Consistency means different instances of the same sub-word model have similar characteristics and trainability means that each speech unit can be trained on sufficiently many examples. Sub-word units can be mono-phones units, syllables, demi-syllables, bi-phones or tri-phones. Mono-phones cannot model the context, whereas others do [18].

Phonemes are sounds that distinguish one word from another in a language and they economically represent speech because most languages have only fifty or fewer phonemes [18]. Tigrigna, for instance, has only 29 phonemes [6]. This implies the fact that vocabulary growth will not necessarily produce the corresponding linear increase in storage and computational demands associated with other methodologies [18]. However, this approach has several limitations one of which is it provides little assistance to handle co-articulation effects. By co-articulation effect, we mean, the effect of the preceding and following sounds in a given utterance. In order to handle this problem, research on

statistical modeling of word segments proposed the use of tri-phones or phonemes-in-context [18]. A tri-phone is a phone that takes its left and right phonetic contexts into consideration [17]. Word models are built up appending the tri-phone models according to a pronunciation dictionary. The dictionary can contain tri-phone or monophone expansions of words.

A syllable, on the other hand, may be defined as a phonological unit of a language with a vowel as its nucleus and the words of all languages are made up of syllables [29]. A Tigrigna syllable may consist of a consonant-vowel or a consonant-vowel-consonant sequence. When three consonants (or one geminated consonant and one simple consonant) come together within a word, the cluster is broken up with the introduction of an epenthetic vowel /ə/, and when two consonants (or one geminated consonant) would otherwise end a word, the vowel /i/ appears after them, or (when this happens because of the presence of a suffix) ə is introduced before the suffix. For example,ክብዲ käbdi 'stomach', ልቢ ləbbi 'heart' -äy 'my', ክብደይ käbdäy 'my stomach', ልቢይ ləbbäy 'my heart' -ka 'your (masc.)', ክብድኻ käbdəxa 'your (masc.) stomach', ልብኻ ləbbəxa 'your (masc.) heart' -n...-n 'and', ክብድን ልብን käbdən ləbbən 'stomach and heart'.

Stress is neither contrastive nor particularly salient in Tigrigna. It seems to depend on gemination, but it has apparently not been systematically investigated. However, languages differ in what comprises a syllable. The use of a syllable as the unit for recognition has two advantages (CSTR, 1999):

1. Syllables exhibit far less context dependencies than phones; and
2. Groups of features (such as voicing, frication, etc.) are not rigidly aligned at phone boundaries, and can spread into neighboring segments. Therefore, it is believed that a syllable model can represent this feature overlap in a more natural way than phone models. One basic limitation of syllables as basic units of recognition is that there are much more syllables than there are phoneme in a language. This enforces trainability problems. This leads to the end of this chapter on the Tigrigna language the phonology and writing system. The next chapter is a discussion on speech recognition and language models.

CHAPTER THREE

LITRATURE REVIEW ON SPEECH RECOGNITION

3.1. Introduction

In automatic speech recognition (ASR) systems acoustic information is sampled as a signal suitable for processing by computers and fed into a recognition process. The output of the system is a hypothesis for a transcription of the utterance. Speech recognition is a complicated task and state-of-the-art recognition systems are very complex. There are a number of different approaches for the implementation of the components [1] and we try to assess them in this chapter.

3.2. Components and Principles of Automatic Speech Recognition

Speech recognition systems have many interrelated components. Figure 3.1 shows the main components of an ASR system. In the first step, the Feature Extraction, the sampled speech signal is parameterized. The goal is to extract a number of parameters ('features') from the signal that has a maximum of information relevant for the following classification. That means features are extracted that are robust to acoustic variation but sensitive to linguistic content. Put in other words, features that are discriminate and allow distinguishing between different linguistic units (e.g., phones) are required. The features should also be robust against noise and control factors that are irrelevant for the recognition process (e.g., the fundamental frequency of the speech signal). The number of features extracted from the waveform signal is commonly much lower than the number of signal samples, thus reducing the amount of data. The choice of suitable features varies depending on the classification technique.

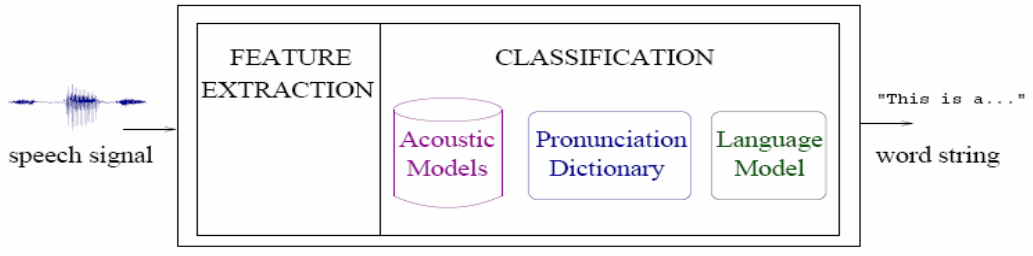


Figure 3.1 Principal of component of ASR System adapted from [41].

Feature vectors are derived from the speech signal. Typically, a frequency-domain based parameterization is performed to extract the features. Spectral analysis is performed, e.g., every 10 ms on the speech samples in a window of, e.g., 32 ms length. The speech signal is regarded stationary in this time-scale. Although this is not strictly true, it is a reasonable approximation. For each frame a vector of parameters, the feature vector, is determined and handed to the next stage, the classification [1].

In the classification module the feature vectors are matched with reference patterns, which are called Acoustic models. The reference patterns are usually Hidden Markov Models (HMMs) trained for whole words or, more often, for phones as linguistic units. HMMs cope with temporal variation, which is important since the duration of individual phones may differ between the reference speech signal and the speech signal to be recognized. A linear normalization of the time axis is not sufficient here, since not all phones are expanded or compressed over time in the same way. For instance, stop consonants ("d", "t" "g", "k", "b", and "p") do not change their length much, whereas the length of vowels strongly depends on the overall speaking rate [42].

The pronunciation dictionary defines which combinations of phones give valid words for the recognition. It can contain information about different pronunciation variants of the same word.

The language model contains rudimentary syntactic information. Its aim is to predict the likelihood of specific words occurring one after another in a certain language. In a more formal description, the probability of the k^{th} word following the $(k-1)^{th}$ previous words are defined as [1]:

$$P(w_k|w_{k-1}, w_{k-2}....., w_1) \quad \ldots\ldots 3.1$$

In practice the context (number of previous words considered in the model) is restricted to $(n-1)$ words

$$P(w_k|w_{k-1}, w_{k-2}... w_1) \approx P(w_k|w_{k-1}, w_{k-2}... w_{k-n+1}) \quad \ldots\ldots 3.2$$

and the resulting language model is called n-gram model

3.3. Types of Speech Recognition Systems

Developing a speech recognition system is a challenging task. There is a need to investigated and address the question of what factors influence the success or failure of a speech recognition system and dictate the degree or sophistication necessary in the design of the system. These factors are should be consider in order to answers to the following questions:

1. Is the system required to recognize a specific individual or multiple speakers?
2. What is the size of the vocabulary?
3. Is the speech to be entered in discrete units (usually words) with distinct pauses among them or as a continuous utterance?
4. Is the system to be operated in a quiet or noisy environment, and what is the nature of the environmental noise if it exists?
5. What are the linguistic constraints to be placed upon the speech, and what linguistic knowledge to be built into the recognizer?

Besides the success of few current systems, they are never without the imposition of one or more of the following constraints [17].

- Utterance.
- Discrete vs. Continuous speech.
- Speaker-dependent vs. Speaker- independent.
- Context-sensitive vs. Context- free.
- Large vocabulary vs. Small vocabulary.
- Word level recognition and Phone level recognition.

3.3.1. Utterance

An utterance is the vocalization (speaking) of a word or words, which represent a single meaning to the computer. Utterances can be a single word, a few words, a sentence, or even multiple sentences [51].

3.3.2. Discrete Speech Vs Continuous Speech Recognition Systems

Discrete speech recognition systems are systems that require the speaker to pause briefly between words [47]. They operate on a single word at a time. The beginning and end points are easier to find and the pronunciation of a word tends not to affect others. The pause is required to help the processor identify word boundaries and to prevent crossword co-articulation from distorting the acoustic pattern of the word to be recognized. However , discrete speech is an unnatural way of speaking that many people find it difficult. It also renders little advantage as to improving the desired level of inputting speed, over typing [18].

Speech is said to be continuous when it is uttered as a continuous flow of sounds with no inherent separations between them [18]. It is our recognition of the distinct words in our mind that make us think speech sounds consist of words with clearly marked boundaries. Continuous speech recognition system is more difficult than discrete recognition system for the following reasons [16]:

- Acoustic variability that results from changes in the environment.
- Intra-speaker variability that results from changes in the speaker's physical and emotional state.
- Inter-speaker variability that results from differences in socio-linguistic background, dialect, vocal tract size and shape. As a result, error rates increase drastically from isolated word to continuous speech.
- It is difficult to find the start and end points of words or word in continuous speech without knowledge of the language.
- Co-articulation effects (inter-unit influences) are much stronger in continuous speech, causing the same sound to appear differently in various contexts; and content words (like nouns and verbs) are often emphasized while function words (like articles) are poorly articulated.
- Content words (nouns, verbs, adjectives.) are emphasized, while function words (articles, prepositions, pronouns) are poorly articulated.

- The recognition of continuous speech is also affected by the rate of speech. Fast or slow speech tends to be harder to recognize than normal speech.

3.3.3. Speaker-dependent Vs Speaker-independent Recognition Systems

A speaker dependent system is developed to operate for speaker/speakers that are involved in the training. It uses speech from the target speaker to learn the model parameters of the speakers' voice during training. Such systems are usually easier to develop, cheaper to buy and more accurate, than but not as flexible as speaker independent systems [26].

Speaker dependent systems are useful for some applications. However, they have many problems – the training is inconvenient and time taking to the user; a large amount of processing is required; certain applications may involve only multiple speakers; certain applications, like telephone directory assistance, may not tolerate the training delay; considerable additional storage is needed if each speaker's parameters are to be stored separately; and as a speaker's voice may change over time due to stress, fatigue, sickness, or variations in microphone positioning, the system is in trouble. In addition a speaker-dependent system uses speech samples from the target speaker to learn the model parameters of the speaker's voice [16]. To supply a reference copy of ones voice pattern (called templates), during training one must speak into the system using a microphone, repeating each word several times [26].

Speaker-dependent recognition systems may be classified into two [26]; namely,

- Isolated word system
- Continuous speech system

The simplest and most common type of speaker-dependent recognition system is an ASR developed for Isolated word. Basically, this system recognizes the voice patterns of only one person and words must be spoken in isolation with clear pauses between

them. Speaker-dependent/Continuous speech systems are complex systems that are capable of dealing with continuous speech, such as a single sentence from a trained user. These systems are complex because they must not only recognize the voice patterns produced but also analyze the syntax of the string of words to deduce its meaning.

Speaker independent systems, on the other hand, are capable of recognizing speech from any new speaker. They are designed to be used by any user, with no enrollment or prior training. Such systems in general are said to be most difficult to develop because those applications must recognize large, heterogeneous populations of speakers [18]. However, they are more flexible and natural. Despite many efforts, no recognizer has yet achieved a reasonable accuracy on a large speaker-independent task, compared to the speaker dependent ones. Research and experiences also reveal as a rule of thumb that for the same task, the error rate of speaker-independent systems have been three to five times speaker-dependent ones [16].

There are three general approaches towards speaker independence [16]. The first approach stands on the assumption that it is possible to find invariant parameters from speeches of (expert) readers and if these invariant parameters can be obtained, speaker-independent recognition could be as easy as speaker-dependent recognition. Thus the approach employs knowledge engineering techniques to find perceptually motivated parameters that are relatively invariant between speakers.

The second approach advocates the use of multiple representations for each reference to capture the between-speaker variations. It requires many speakers utter each word in the vocabulary. These multiple examples will be divided in to several clusters, and a prototype will be generated from each cluster. However, both these two approaches have been experimented and produced good results only for very limited tasks, but has not been successfully extended to large-vocabulary tasks[17].

The final approach is speaker adaptation. It begins with an existing set of parameters and a small number of adaptation sentences from a new speaker. These sentences are used to

modify the parameters so that they are adjusted to the new speaker. However, the adaptation process should be rapid non-intrusive; otherwise it degenerates to speaker dependent systems. Solomon is working on speaker adaptation for the syllable-based model and tying to improve his speech recognizers [37].

Speaker-independent systems, too, can be classified into two [26]; namely,

- Speaker-independent /isolated word systems
- Speaker-independent /Continuous speech systems

Speaker-independent /Isolated word systems are intended for virtually anyone without enrollment. The vocabulary is limited; typically only ten digits and fewer words. Systems of these kinds are suitable for a large number of people using a restricted vocabulary.
Speaker-independent /Continuous speech systems are the ultimate goal of speech recognition research. However on account of the many difficulties involved in continuous speech recognition, it is generally accepted that the present situation makes only fairly restricted systems feasible [26].

3.3.4. Context Sensitive vs. Context Free Systems

When speech is produced as a sequence of words, language models or artificial grammars can be used to restrict the permissible combination of words. More general language models approximating natural language are specified in what is known as a context-sensitive grammar [46]. Speech recognition systems that increase their accuracy by anticipating or limiting what can be said at any given time are referred to as context-sensitive recognition systems whereas systems that allow users to say anything, anytime without any constraint are called context free recognition systems. Context-sensitive recognition systems are more difficult to implement on computers than context-free ones [46].

3.3.5. Large Vocabulary Systems vs. Small Vocabulary Systems

Vocabularies (or dictionaries) are lists of words or utterances that can be recognized by the speech recognition system. The size of vocabulary of a speech recognition system affects the complexity, processing requirements and the accuracy of the system. Though there are no established definitions, [50] provide distinctions shown in the table 3.1

Small vocabulary	tens of words
Medium vocabulary	hundreds of words
Large vocabulary	thousands of words
Very large vocabulary	tens of thousands of words

3.1 Tables: Different classification of vocabularies size.

Some applications such as telephone dialing system and most manufacturing applications only require a few words (e.g. numbers only) [18]. Large vocabulary systems are very attractive for applications such as dictation system and may be poorly suited to smaller-vocabulary command and control applications [29].

Generally, smaller vocabularies are easier for a computer to recognize, while large vocabularies are more difficult [43]. As the vocabulary size gets larger, the number of confusable words grows substantially. Besides, with small vocabularies, each word can be modeled individually, because it is reasonable to expect sufficient training for the handful of words.

It is also possible to store the parameters of each word model separately. However, in large vocabularies it is difficult to train each word explicitly because neither the training nor the storage is available. The other problem relates to the complexity of search. For small vocabularies, it is possible to perform optimal search.

3.3.6. Word level recognition vs. Phone level recognition

Word level recognition system recognizes each word independently. There is a front-ends model created for every word, and is trained by the training set which contains that word spoken by many different speakers. For each new word to be added, a training set needs to be created. Word level recognition is useful to be recognized with very small number of words [33].

Phone level recognition system brakes up each word into phones which are recognized independently. This is the more useful model which can be used for continuous speech recognition. However a phonetic dictionary is required for phone level recognition. For the word level recognition, we need a training set which is a set of speech file recordings of the same word by different speakers [33].A front-ends model is trained for each of the vocabulary words using the training set, and it is tested using test data set ,a data set different from the training data. The percentage of times the word is recognized correctly is a measure of the performance of the recognizer [33].

3.4. Signal processing

The first element of all ASR system is signal processing, and a signal processing can be modeled as be a series of piece wise stationary signals, which can be decoded to a set of one or more symbols. The continuous speech waveform is thus first converted into a sequence of discrete parameter vectors called speech vectors. It is assumed that for the duration of the vector (typically 10ms) speech can be regarded as a stationary process [27].

The basic goal is to "decode" the message and then convert it either into writing (e.g. dictation machine) or into commands to be processed (e.g. hands free dialing). The role of the speech recognition system is to create a mapping between the speech vectors and the corresponding symbols. However since the waveform is not always the same, due to its dependency in the speaker and the time, the mapping is not one to one [27].

The boundaries between the symbols corresponding to the word are also not known, which is challenging for speech recognizer signal processing, commonly used as front end analysis converts, the speech wave form in to some type of parametric representation for further analysis and processing [27],is the short time spectral envelope .The most common important parametric representation of speech is by this method, the feature vectors used to characterize the spectral properties of the input speech are derived.

There are two dominant methods of spectral analysis: Linear predictive coding (LPC) and Mel-Frequency Cepstral Analysis [27].The most widely used decoder is implemented by Linear Predictive Coding (LPC) technique [29]. Linear Predictive Coding (LPC) has become the dominant coding methodology for speech recognition. According to Markowitz in linear prediction (LP) analysis, the vocal tract transfer function is modeled by an all-pole filter with transfer function.

$$H(z) = \frac{1}{\sum_{i=0}^{p} a_i z^{-i}} \quad \text{.....3.3}$$

Where p is the number of poles and $a_0 = 1$. The filter coefficients $\{a_i\}$ are chosen to minimize the mean square filter prediction error summed over the analysis window. The auto correlation method can perform this optimization as follows.

Given a window of speech samples $\{s_n, n = 1, N\}$, the first 1 + p terms of the auto correlation sequence are calculated from

$r_i = \sum_{j=1}^{N-i} s_j s_{j+1}$ 3.4

Where i = 0, 1, 2 … p. The filter coefficients are then computed recursively using a set of auxiliary coefficients $\{k_i\}$, and the prediction error E which is initially equal to r_0

let $\left\{k_j^{(i-1)}\right\}$ and $\left\{a_j^{(i-1)}\right\}$ be the reflection and filter coefficients for a filter of order 1 − i , then a filter of order can be calculated in three steps. Firstly, a new set of reflection coefficients are calculated.

$k_j^{(i)} = k_j^{(i-1)}$ 3.5

For j = 1, 1 − i

$$k_j^{(i)} = \left\{r_i + \sum_{j-1}^{i-1} a_j^{(i-1)} r_{i-j}\right\} / E^{(i-1)} \quad \text{.....3.6}$$

Secondly, the prediction energy is updated.

$$E^{(i)}=\left(1-k_i^{(i)}k_i^{(i)}\right)E^{(i-1)}$$ ……..……3.7

Finally, new filter coefficients are computed

$$a_j^{(i)}=a_j^{(i-1)}\_k_i^{(i)}a_{i-j}^{(i-1)}$$ …….3.8

For j = 1, 1 − i $a_j^{(i)}=-k_i^{(i)}$ ……..3.9

This process is repeated from i=1 through to the required filter order p = i . For speech recognition purpose we can choose LP coefficients $\{a_i\}$ or LP reflection coefficients $\{k_i\}$ by going through the above transformation. The required filter order must be set for use in speech recognition, which is the speech parameter vector size [18].

On the other hand, MFCC standing for Mel-Frequency Cepstral Coefficients (MFCCs) deals with power spectrum of a speech signal which describes the frequency content of the signal over time [3]. The purpose of MFCC is to reduce the number of data characterizing the signal and shows a limited number of parameters or coefficients, discriminating and robust [39].These are calculated from the log filter bank amplitudes {Mj } using the Discrete Cosine Transform.

$$C_i=\sqrt{\frac{2}{N}}\sum_{j=1}^{N}m_j\cos\left(\frac{\pi i}{N}(j-0.5)\right)$$ ……..…….. …….. 3.10

Where N is the number of filter bank channels. The required number of cepstral coefficients is set by the maximum value of the iteration made which are mostly 13.
This is done by a known mathematical function called Discrete Fourier Transform (DFT) [41], which computes the frequency information of the equivalent time domain signal. Discrete Fourier Transform (DFT) is used instead using the Fourier Transform during analyzing speech signals. It is because the speech signal is in the form of discrete number of samples due to preprocessing. The input of the DFT is a windowed signal x[n]…x[m], and the output, for each of N discrete frequency bands, is a complex number X[k] representing the magnitude and phase of that frequency component in the original signal. The discrete Fourier Transform is represented by the

equation below, where X (k) is the Fourier Transform of x (n).Mathematical details of DFT, includes the note of Fourier analysis, also relies on Euler's formula stated below

$$X[k] = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x[n] e^{-j\frac{2\pi}{N}kn}, \qquad 0 \le k \le N-1$$

……..3.11

However, as a speech signal contains only real point amplitude values, a real-point Fast Fourier transform (FFT) is an optimized version of the Discrete Fourier Transform [41]. It takes a window of size 2^k and returns a complex array of coefficients for the corresponding frequency curve. In feature identification, the frequency characteristics of the speech information can be considered as a list of "features" for that speaker. If we combine all windows of the voice sample by taking the average between them, we can get the average frequency characteristics of the sample. Subsequently, if we average the frequency characteristics for samples from the same speaker, we can essentially find the center of the cluster for the speaker's samples. Once all speakers have their cluster centers recorded in the training set, the speaker of an input sample could be identified by comparing its frequency analysis with each cluster center by using some classification method [41]. Though its recognition rate is higher than LPC algorithm, the run time of the recognize process is too much which can't satisfy the real-time system's demand [8].

3.5. Acoustic model

The core of an acoustic model lies in the capabilities of the feature vectors to capture the distinctive properties of the speech. A good acoustic model should take into account speaker variations, pronunciation variations, environmental variations and context dependent phonetic co-articulation variations. For this reason, the acoustic training corpus has to be quite large to obtain a robust acoustic model. The underlying assumption behind any recognition system is that the waveform of a speech signal that comes out of a speaker's vocal apparatus is a realization of the concept that was in the form of symbols in his/her mind. When a source conceives an idea to speak out, it was understood symbolically. The moment it gets out to the channel, it materializes in the form of speech signals or sound waves. Thus, one direct and possible approach for a

computer based speech recognition system to recognize an utterance is inferring the original symbols from the speech signals [43].

To effect this reverse operation of recognizing the underlying symbol sequence given a spoken utterance, first the speech waveform should be digitized and important features must be extracted from the digitized format. Then the extracted features out of the continuous speech waveform should be converted to a sequence of equally spaced discrete parameter vectors [43].

The next requirement for a computer-based speech recognizer is to make estimation of the possible symbol sequences that would result in information carried by the sequence of vectors. Acoustic models can help such speculations about the probability of the observed symbol sequences given the models. Then a language model like n-gram can be used to formulate and tell the prior probability of estimated word sequences. The final role of the recognizer is to affect a mapping between sequences of speech vectors and the wanted underlying symbol sequences. Modeling techniques and matching algorithms will be used to find out the required association between hypothesized sequences of words and the uttered speech. This module is referred to as acoustic modeling or statistical pattern recognition or acoustic pattern matching. One important approach of acoustic modeling, these deep is Hidden Markov Modeling which is fundamentally probabilistic. It measures the association of utterances to reference patterns by the application of statistical models [21].

In Hidden-Markov-based acoustic modeling, the observation probabilities of the vectors extracted feature vectors of the speech are computed. This probability is represented by:

$$P(o_t \setminus w_j) \quad \text{……..3.12}$$

Where o_t: vector observed at time t_w

j: j^{th} vocabulary word

During search or decoding the system finds the right word or words uttered among all words in the vocabulary in an efficient way. To find the most likely word or word sequence, the ASR system searches a network of words. The size and the shape of the

search space are mainly determined by the language model. Language model involves the utterance probability of words and depends on the subject spoken about. Language model probability is represented by $P(w_j)$.In fact; an ASR system tries to maximize the probability that the uttered word is w_j given the observation vector sequence O,

$$\text{argmax}_j \; [P(w_j \backslash O)] \quad \text{3.13}$$

$$\text{where } O = o_1 \; o_2 \; o_3 \; \ldots . \; o_{t-1} \; o_T$$

This probability is not computable directly but using Bayes' rule gives

$$P(w_j \backslash o) = \frac{P(o \backslash w_j) \; P(w_j)}{P(o)} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 3.14$$

For an isolated word recognition (IWR) task, it is sufficient to compute (3.14). For a CSR system, the formula below takes the form;

$$P(W \backslash O) = \frac{P(O \backslash W)(P(W)}{P(O)} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 3.15$$

$$W = w_1 \, w_2 \, w_3 \ldots\ldots w_{Q-1} \; w_Q \quad Q \leq N$$

Where N is the number of words in the vocabulary, The probability P(W) includes the syntactic and semantic conditions together. If only the syntactic conditions are held, the language model becomes a grammar. This type of language models are generally represented in the form of finite state networks.

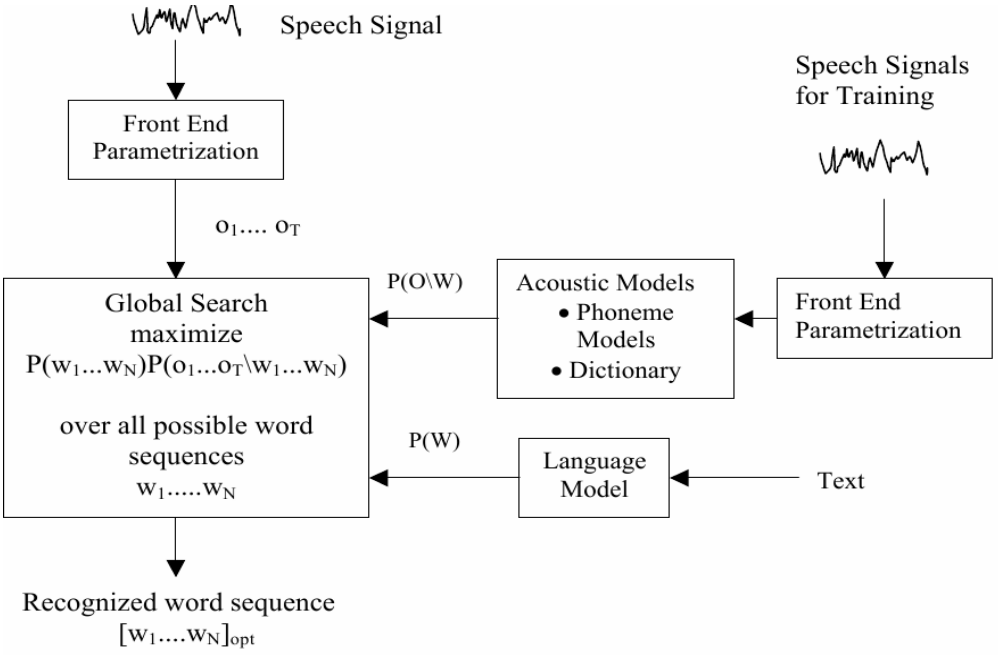


Figure 3.2 Overview of a statistical speaker independent CSR system

A general demonstration of the structure of a statistical CSR system can be seen in figure 3.2[31]. Figure 3.2 shows the computation of the probability P(W\O). The prior probability P(W) is determined directly from a language model by analysis the text corpus. The likelihood of the acoustic data P (O\W) is computed using a composite HMM representing W constructed from simple HMM phone models that are joined in sequence according to word pronunciations stored in a dictionary. The next section is about the acoustic model, namely obtaining the feature vectors and obtaining the observation probability P (O\W).An ASR system is has a Front end and back end analysis. In front end part, the feature vectors of the speech signal are extracted. In the back end, the actual recognition takes place [31].

3.5.1. Front-end Analysis or process

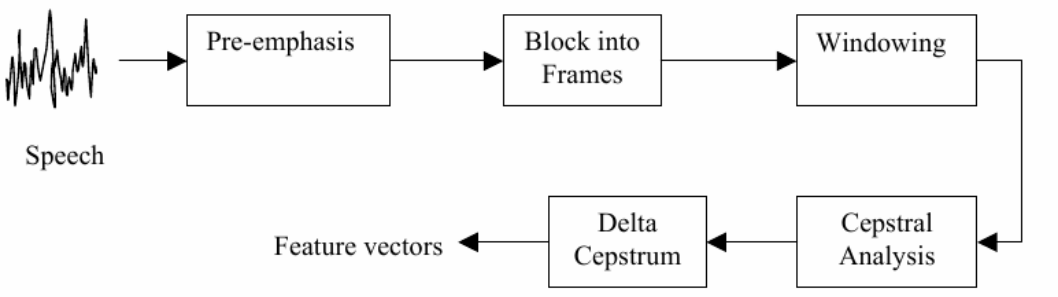
Front-end analysis refers to the first stage of ASR, whereby the input acoustic signal is converted to a sequence of acoustic feature vectors. The short-term spectrum provides a convenient way of capturing the acoustic consequences of phonetic events. Ideally the method of front-end analysis should preserve all the perceptually important information for making phonetic distinctions, while not being sensitive to acoustic variations that are irrelevant phonetically[42].

ASR seems desirable not to use features of the acoustic signal that are not used by human listeners, even if they are reliably present in human productions, because they may be distorted by the acoustic environment or electrical transmission path without causing the perceived speech quality to be impaired. Over the years many different front-ends have been tried, for use first with DTW recognizers and, more recently, with HMM systems [13].

These front-ends vary in the extent to which they incorporate knowledge about human auditory perception, but currently the most successful analysis methods include at least some of the known properties of perception. These successful methods are, however, also characterized by a compatibility with the mathematical techniques that are generally used in HMM recognizers [7]. The spectrum of voiced speech is characterized by a

downward trend, whereby frequencies in the upper part of the spectrum are attenuated at 6 dB. This downward trend is due to a combination of the typical -12 dB slope of the glottal source spectrum with the +6 dB/octave lift given by the radiation effect due to the lips. For the purpose of front-end analysis, it is common to compensate by applying a pre-emphasis of 6dB so that the analyzed signal has a roughly flat spectral trend.

This pre-emphasis is easily applied to the speech signal as the first processing stage. Although the above argument for pre-emphasis only applies to voiced regions, in practice it is usually applied throughout without causing any obvious problems for the analysis of voiceless regions. A block diagram of a front end processor can be seen in Figure 3.3 [21]. Today's systems still cannot match human's performance. In spite of construction of a very accurate speech recognizer for a particular speaker, in a particular language and speaking style, in a particular environment and limited to a particular task, it remains as a problem to build a system that can understand anyone's speech, in any language, on any topic, in any fluent style and in any environment. The frequency range of human voice does not include informative components at frequencies higher than 5 kHz, so it is reasonable to reduce the amount of high frequency noise by low pass filtering. Moreover, the acoustic transfer function of human ear balances the spectrum before perception.



Finger 3.3 Block diagram on front end processor.

Due to physical constraints, the vocal tract shape generally changes fairly slowly with time and tends to be fairly constant over short intervals (around 10–20 ms). A reasonable approximation is therefore to analyze the speech signal into a sequence of

frames, where each frame is represented by a single feature vector describing the average spectrum for a short time interval. The frames have to be multiplied by a window function (windowing), so that the transitions from frame to frame can be smoothed. Windowing is also beneficial to recover all parts of the signal and to eliminate possible gaps between frames. Without windowing, the spectral envelope has sharp peaks and the harmonic nature of a vowel is not apparent. The window function used in this thesis is Hamming window [13]. Its formula is given by:

$$W(n) = 0\,54 - 0.46\cos(2\pi(n/N-1)) \quad \text{.....3.16}$$

Where, N is the length of the frame in terms of samples.

This type of windowing is necessary to reduce any discontinuities at the edges of the selected region, which would otherwise cause problems for the subsequent frequency analysis by introducing spurious high-frequency components into the spectrum. The length of each analysis window must be short enough to give the required time resolution, but on the other hand it cannot be too short if it is to provide adequate frequency resolution. In addition, because the analysis is normally performed at a fixed time interval, during voiced speech the window must be long enough so that it is not sensitive to exact position relative to the glottal cycle (i.e. there needs to always be at least one complete glottal cycle in the main part of the window[44].

Long windows also have the advantage of smoothing out some of the random temporal variation that occurs in unvoiced sounds such as fricatives, but at the expense of blurring rapid events such as the releases of stop consonants. A common compromise is to use a 20–25 ms window applied at 10 ms intervals giving a frame rate of 100 frame or frames and an overlap between adjacent windows of about 50%.

In the next step, the Fourier transformed signal is passed through a set of band passes filters that have triangular shape. The bandwidths (critical band) and center frequencies are determined by experimental results on human hearing. The Mel-frequency scale is designed to approximate the frequency resolution of the human ear. The frequency resolution of the human ear is linear up to 1000 Hz and logarithmic thereafter. This means that the band pass filters get larger at higher frequencies.

The log filters outputs are then used to obtain the cepstral coefficients with a discrete cosine transform formula:

$$C\ i=\sqrt{2}/n\sum mj\cos(i\pi/N(\ j-0.5) \quad \text{..... 3.17}$$

Where N is the number of filter bank channels and m_j is the output of the j^{th} filter.
This has the effect of compressing the spectral information into lower order coefficients and it also de-correlates them. De-correlation allows the statistical modeling to use diagonal covariance matrices [43].

The acoustic model assumes that each feature vector is uncorrelated with its neighbors. This is a rather poor assumption, because the physical structure of the human vocal apparatus ensures that there is continuity between successive spectral estimates [42]. To capture the changes in the signal, a feature vector used to describe the signal for one frame can additionally contain features of neighbor frames or the difference between features of the predecessor and successor frames. Furthermore, the difference of the differences can be added to the feature vector, too. So, a feature vector consists of cepstral and an energy coefficient length, by concatenating the delta and acceleration cepstral and energy coefficients [43].

3.5.2. Sampling

The most commonly used sampling technique is based on the Nyquist sampling theory [29]. The Nyquist theorem states that if an analog signal is sampled at regular intervals at a rate at least twice the highest frequency in the channel, the samples will contain sufficient information about the signal to allow its reconstruction. The sampling rate is the number of points observed per second. Most speech recognition preprocessors take 8,000 to 10,000 samples per second converted into a sequence of parameter vectors at a certain frame rate of mostly 10ms, 15ms, and 20ms [18]. In statistical automatic speech recognition, the speech waveform is sampled at a rate between 6.6 KHz and 20 KHz [8]. No matter what the sampling rate is, the speech signal must be suitably low-pass filtered prior to the sampling process to eliminate undesired high frequencies of the speech and high frequency noise [15].A filter takes sound of

any frequencies as input and outputs sound only in certain frequency ranges. A low-pass filter with a cutoff of N Hz outputs only frequencies of N Hz or less. A high-pass filter with the same cutoff outputs only frequencies of N Hz or greater. There are also band-pass filters that output frequencies within a certain predefined [29].

3.5.3 Dynamic Time Warping (DTW) and Hidden Markov Models (HMM)

3.5.3.1 Dynamic Time Warping (DTW)

Using DTW, compare two sequences of vectors: reference R = [r(1),...r(R)] of the length R and test O =[o(1),...o(T)] of the length T [13]. The aim is to calculate the distance between them D(O,R). In the simplest case, the i-th word is represented by the word w_i in the vocabulary R_i, the recognizing word is given thus as:

$$i^* = argmin_i D(O, R_i) \quad \text{……..3.18}$$

During DTW, define a matrix D of the size T × R, that is fill with the local distances between the vectors :d(i,j) = d(o(i),r(j)). Distance d (·,·) is here the cepstral measure. Another important matrix is G – matrix of the partial cumulated distances. Compared to D, matrix G has zeros row and zeros column, that are initialized with the values ∞, except the cell g (0,0) = 0. For the local restriction of the path, type I, and for the type weights a. we can calculate next items G:

$$g(i,j) = min \begin{cases} g(i-1,j) + d(i,j), \\ g(i,j-1) + d(i,j) \\ g(i-1,j-1) + 2d(i,j) \end{cases} \quad \text{…….3.19}$$

for 1 ≤ i ≤ T and 1 ≤ j ≤ R. The last matrix item g(T,R) presents the optimal distance:

$$D(O,R) = \frac{g(T,R)}{T+R} \quad \textbf{…….3.20}$$

In matrix **G** we can find the optimal path backwards. All these operation are implemented in the function **dtw.m.** Look at the function carefully. The function uses the DTW distance as the input. The graphical output of the function is realized [13]:

- The upper panel presents matrices of the local distances (vectors “all by all”).

- The middle panel presents matrices of partial cumulated distances.
- The bottom panel presents the path. The panel title presents the DTW distance.

3.5.3.2. Hidden Markov Models (HMM)

HMM recognizer is model characterized by [13]:

- N states, only N − 2 are "emmiting", the first and the last one are used only for convenience
- Left-to-right structure. Matrix of transition probability has the nonzero items only on the diagonal.
- The emmition probabilities are modeled using P-dimensional Gaussian distribution:

$$b_j[\mathbf{o}(t)] = \mathcal{N}(\mathbf{o}(t); \mu_j, \mathbf{\Sigma}_j) = \frac{1}{\sqrt{(2\pi)^P |\mathbf{\Sigma}_j|}} e^{-\frac{1}{2}(\mathbf{o}(t)-\mu_j)^T \mathbf{\Sigma}_j^{-1}(\mathbf{o}(t)-\mu_j)}, \quad \ldots$$

3.21

As the input sequence O LPC-cepstral vectors are used without c_0 same as for DTW.

3.5.3.2.1 Types of HMMs

we have only considered the special case of ergodic or fully connected HMMs, in which every state of the model could be reached (in a single step) from every other state of the model.(Strictly speaking an ergodic model has the property that every state can be reached from every other state in a finite a number of steps).

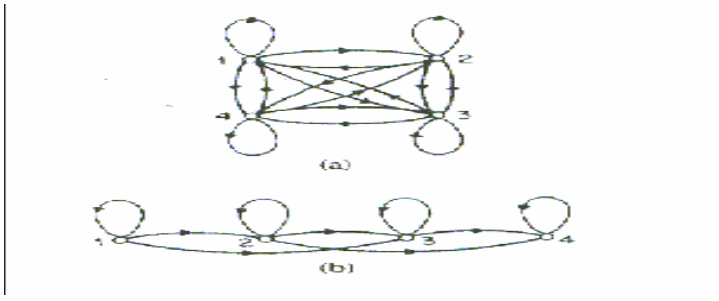


Figure 3.4 Illustration examples of types of HMMs.

a) State ergodic model Vs. 4-state left-right model

As shown in Figure 3.4 (a), for an N = 4 state model this type of model has the property that every a $_{ij}$ coefficient is positive. Hence, for the example of Figure3.4 (a) we have

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

For some applications, other types of HMMs have been found to account for observed properties of the signal being modeled better than the standard ergodic model. One such model is shown in Figure 3.4 (b). This model is called a left-right model or Bakis Model because the underlying state sequence associated with the model has the property that as time increases the state index increases (or stays the same), i.e. the states proceed from left to right. Clearly the left-right type of HMM has the desirable property that it can readily model signals whose properties change over time for example Speech. The fundamental property of all left-right HMMs is that the state transition coefficients have the property

$a_{ij} = 0, j < i$ …….(3.21)

i.e. no transitions are allowed to states whose indices are lower than the current state. Furthermore, the initial state probabilities have the property

$$\pi_i = \begin{cases} p, & i \neq 1 \\ 1, & x = 1 \end{cases}$$ 3.22

Since the state sequence must begin in state 1 (and end in state N). Often, with left-right models, additional constraints are placed on the state transition coefficients to make sure that large changes in state indices do not occur; hence a constraint of the form is often used.

$a_{ij} = 0, j > i + \Delta$ …….3.23

In particular, for the example of Figure 3.4 (b), the value of _ is 2, i.e., no jumps of more than 2 states are allowed. The form of the state transition matrix for the example of Figure 3.3 (b) is thus:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix}$$

It should be clear that, for the last state in a left-right model that the state transition coefficients are specified as

$a_{NN} = 1$……..a

$a_{Ni} = 0$, for i < N……..b

It should be clear that the imposition of the constraints of the left-right model, or those of the constrained jump model, essentially have no effect on the re estimation procedure.

b) Discrete HMMs vs. Continuous HMMs

On the other hand HMMs may also be catagorized based on the type of output probability distribution $b_j(O_t)$. Accordingly, there can be

1. Discrete probability distributions (Discrete HMMs)
2. Continuous output probability density functions (Continuous HMMs and tied-mixture HMMs)

Among these variants continuous-density HMM and the tied-mixture HMM, are two of the most popular types of HMM in current ASR systems [8]. Such HMMs most commonly use a multivariate Gausian output distribution1 [3]. HTK is also defaulted to continuous density systems in preference to discrete systems because, for the purpose of research, they have a number of mathematically desirable properties and, for the purpose of practical application; they are believed to be more robust [43].

In discrete HMMs, the observations are characterized as discrete symbols chosen from a finite alphabet, and therefore a discrete probability density with in each state of the model can be used. The problems with this approach are: each observation is associated with one output only it doesn't reflect correlation between two discrete units when spoken in continuous speech. To use a continuous observation density, some restrictions must be placed on the form of the model probability density function (pdf) to ensure that the parameters of the pdf can be re-estimated in consistent way. The most general

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

$$b_j(o) = \sum_{k=1}^{M} C_{jk}\, N(o, \mu_{jk}), 1 \le j \le N$$ …….3.24

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

HTK is designed primarily for modeling continuous parameters using continuous density multivariate output distributions [43]. It can also handle observation sequences consisting of discrete symbols in which case, the output distributions are discrete probabilities. In continuous density HMMs each observation probability distribution is represented by a mixture Gaussian density.

3.5.3.2.2 Limitations of HMMs

Although use of HMM technique has contributed greatly to recent advances in speech recognition there are some inherent limitations of this type of statistical model for speech. A major limitation is the assumption that successive observations (frames of speech) are independent, and therefore the probability of a sequence of observations P(O_1, O_2 ... O_T) can be written as a product of probabilities of individual observations, i.e.,

$$P(O_1, O_2 \ldots O_T) = \prod_{i=1}^{N} P(o_i)$$ ……. …… 3.25

Another limitation is the assumption that the distributions of individual observation parameters can be well represented as a finite mixture of Gaussian densities. Finally the Markov assumption it self, i.e., that the probability of being in a given state at time t only depends on the state at time t - 1, is clearly inappropriate for speech sounds where dependencies often extend through several states. However, in spite of these limitations this type of statistical model has worked extremely well for speech recognition problems.

3.6. Language Modeling

Small vocabulary recognition generally do not rely heavily on language models to accomplish their tasks because they are mainly used in command and control applications where the vocabulary words are essentially acoustic control signals that the vocabulary has to respond by.

Some language models may be designed in such a way that their finite state grammars produce the same set of sentences with the intended task. They are committed to the sentences in the application or the task. Such grammars will be too simplistic and have low perplexity. However, the long-range goal of many systems is and should be to ensure high perplexity (looser grammar) and that supports large vocabulary [17].

Some other language models may be of simple grammar that specifies only the list of words that can legally follow any given word. Such model is referred to as word pair grammar. In using this grammar, all the HMMs for all the words in the vocabulary will be put in parallel. A null transition will be used from the last state of a word, say X, to the first state of the other word, say Y, if (X, Y) is a legal/possible word pair. The transition probability of this null is 1/N, where N is the number of words in the vocabulary, which assigns, probability for all transitions of legal pairs [27].

A speaker independent continuous speech recognition system, however, is generally dependent on linguistic knowledge. Hence, incorporation of knowledge of the language, in the form of a language model, P (O\W), is essential for continuous speech recognition systems [28]. The probability P (W) has a great effect on search process. It provides effectiveness and efficiency since it does not depend on the acoustic data, only a text corpus is needed to compute. Language models are used to model regularities in natural language. Such language models may incorporate syntactic or semantic constraints. Often, when only syntactic constraints are used, the language model is called a grammar. Usually, they are represented in a finite state network so as to be integrated into the acoustic model in a straight forward manner [27].

The most popular methods are statistical n-gram models, [42] which attempt to capture the syntactic and semantic constraints by estimating the probability of a word in a sentence given the preceding (n-1) words. Hence, this probability strictly depends on the amount of the text corpus and its subject.

3.6.1. N-grams

In n-gram language model there is a logical assumption that the priori probability of the word sequence, $W = w_1 \ldots\ldots w_N$ can be computed as;

$$P(W) = P(w_1)\, P(w_2 \setminus w_1)\, P(w_3 \setminus w_1\, w_2) \ldots\ldots P(w_N \setminus w_1\, w_2 \ldots\ldots w_{N-1}) \ldots..3.26$$

It is impossible to obtain this conditional word probability for all possible word sequences of different lengths. Therefore, N-gram models, such as bigrams and trigrams are used [40]. The term can be approximated as [39];

$$P(W) \approx P(w_j \setminus w_{j-N+1} \ldots\ldots w_{j-1}) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 3.27$$

The value of N is a trade-off between the stability of the estimate and its appropriateness. A trigram (N=3 is a trigram word model in which probabilities of sequences of 3 words in a specified order are given) is a common choice with large training corpus whereas a bigram (N=2 is a bigram word model in which probabilities of sequences of 2 words in a specified order are given) is often used with smaller ones. As the amount of N gets larger, it gets more difficult to reliably estimate the priori probability. This probability can be estimated by relative frequency approach.

$$P(w_i \setminus w_{i-N+1} \ldots\ldots w_{i-1}) = \frac{F(w_{i-N+1} \ldots\ldots w_{i-1}\ w_i)}{F(w_{i-N+1} \ldots\ldots w_{i-1})} \qquad \ldots\ldots\ldots\ldots\ldots 3.28$$

Where, F is the number of occurrences of the string in its argument in the given training corpus.It is obvious that possible word sequences may not be observed in the training corpus. This means, a zero probability is appointed to the unseen N-grams. In addition, distribution function of the frequencies may have sharp edges, i.e. some of the words may occur lots of times whereas others may occurs only a few times ,Instead of

straightforward estimation from counts, various smoothing techniques have been proposed to balance the probabilities[42]. These include discounting the estimates, recursively backing-off to lower N-grams and interpolating N-grams of different order [40]. The most widely used once in speech recognition system is the back-off n-grams. In back-off smoothing, the N-grams that are not observed or the number of occurrences of which are below a threshold during training are assigned a nonzero probability related to the unigram probabilities. Assigning all strings a nonzero probability helps prevent errors in speech recognition. This is the core issue of smoothing.

There are other back-off techniques proposed in the literature such as Kneser-Ney smoothing and Katz smoothing based on Good-Turing estimates. Kneser-Ney method slightly outperforms other smoothing techniques for both bigrams and trigrams [43]. Whenever training data is sparse, smoothing can help performance and sparseness is almost always an important issue in statistical modeling. In Figure3.4, a simple search space based on bigrams is illustrated. There is not a direct connection between w_k and w_j, because in training corpus the word pair (w_k ,w_j) is not observed. Only observed bigrams are connected by direct transitions with correspondent bigram probabilities. This significantly reduces the bi-gram expansion [42].

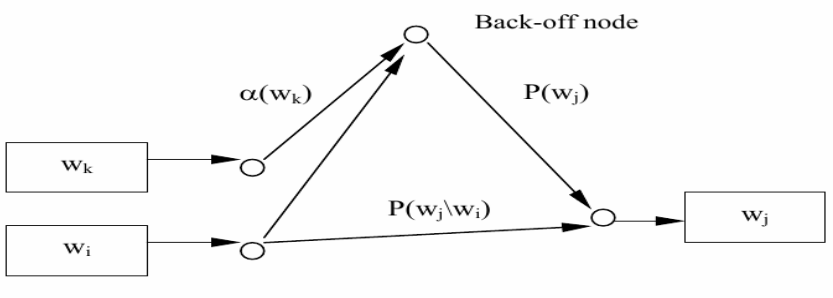


Figure 3.5: Illustration of the back-off node for a bi-gram model

3.7. Related Works

Taking the advantage of speech recognition systems into consideration, some interesting works have been done on speech recognition for the local languages. They have put a milestone to make understand the basic concepts and to introduce different techniques of

speech Recognition. In the following sections, the paper tries to review different works related to speech recognition for local languages using HMM.

3.7.1 Speech recognition systems for local languages

Research in automatic speech recognition for Amharic started in 2001 when Solomon [34] developed a speech recognizer for Amharic that recognizes a sub-set of isolated consonant-vowel (CV) syllable using the HTK (Hidden-Markov Modeling Toolkit). He selected 41 CV syllables of Amharic language out of 234. Speech data of the selected CV syllables has been recorded from eight people (4 male and 4 female) with the age range of 20 to 33 years. The average recognition accuracies achieved were 87.68% and 72.75% for speaker dependent and speaker independent systems, respectively. As indicated by the researcher, the result seems to be low compared to systems developed for other languages. This might be due to problems of the recording environment and insufficient training speech data.

As a continuation, Kinfe[14] developed sub-word based isolated word recognition systems for Amharic using HTK. The sub-word units used in his experiment are phones, triphones, and CV-syllables. He considered 20 phones (out of 39) and 104 CV syllables, which are formed using the selected phones. Speech data of 170 words, which are composed of the selected sub-word units, have been recorded from 20 speakers (speech of 15 speakers for training and the remaining for testing). Speaker dependent phone-based and tri-phone-based systems have an average recognition accuracy of 83.07% and 78% respectively. Phone-based and tri-phone-based speaker independent systems have an average recognition accuracy of 72% and 68.4% respectively. According to Kinfe a comparison of the different sub-word units revealed that the use of CV syllables has led to relatively poor performance. This is a contradiction with recent findings of [37] that attributed the poor performance of Kinfe's CV syllable-based recognizers to the use of insufficient speech data.

Zegaye[45] further, 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 using HTK. For training and testing recognizer, he used the speech data, which amounts to of 8000 sentences read by 80 people, recorded by Solomon et al. (2005). The best recognizer was a tri-phone based recognizer which has 76.2% word recognition accuracy.

According to Molalgne sited by Solomon, Martha and Wolfgang [20] tried to compare HMM-based small vocabulary speaker dependent continuous speech recognizers built using two different toolkits: HTK, and the MSSTATE (it is a Toolkit used for speech recognition produced by Mississippi state university from Mississippi State). He collected a speech data of 50 sentences with ten words (the digits) from a single speaker. The speaker dependent recognition system developed using the HTK has 82.5% word recognition accuracy whereas the system developed using MSSTATE has 79.0% word recognition accuracy. He also found out that HTK was faster than MSSTATE Toolkit.

Hussein and Gambäck [10] developed an Amharic speaker independent continuous speech recognizers based on an HMM/ANN hybrid approach. The recognizer was constructed at a context dependent phone level with the help of the CSLU Toolkit. They used part of the data (5000 sentences) recorded by Solomon [37]. The recognizers were tested with a total of 20 sentences read by 10 speakers (2 sentences each) who also read the training data. The best result obtained with this test data was 74.28% word recognition accuracy. When tested with test data read by speakers who were not in the training, performance degradation was observed. There was 4.28% absolute word recognition accuracy reduction, since there was no usable speech corpus for the development of a large vocabulary speech recognizer.

Solomon [37] developed an Amharic speech corpus that can be used for various kinds of investigations in the development of ASR for Amharic. He explored various possibilities for developing a Large Vocabulary Speaker Independent Continuous Speech Recognition System for Amharic [37].Solomon assumed that, due to their highly regular Consonant Vowel (CV) structure; Amharic syllables lend themselves to be used as a basic recognition unit. Indeed, he has been able to show that syllable models can be used as a

competitive alternative to the standard architecture that is based on tri-phone models. The acoustic model of the syllable-based recognizer requires 15MB memory. Together with the language model and use of speaker adaptation it has a word recognition accuracy of 90.43% on the 5,000 words evaluation test set at a speed of 2.4 minutes per sentence. While the acoustic model of the tri-phone-based recognizer requires 38MB memory and has a word recognition accuracy of 91.31% on the same test set at a speed of 3.8 minutes per sentence. Although this is the state-of-the-art recognition performance in Amharic, there are still rooms for improvement. Solomon mentioned some of the limitations of in the areas of speech corpora, language model, acoustic models, and the application of the recognizers.

Nebiyou [23] attempts to design speech recognition STT (Speech to Text) system, for Amharic language. Amharic language has more than 200 characters but the standard keyboard is made for English alphabet. This limited number of keys has imposed the need of 2 – 4 key strokes to write a single Amharic letter. This thesis is an attempt to build STT (Speech to Text) conversion for Amharic language using the statistical approach. So the inventory of speech files is made by recording and from these data appropriate models are built. The purpose is to test the performance based on the models built and prove that statistical models are suited to modeling speech signals. Statistical approach to speech recognition requires a large amount of recorded speech data (Corpus data). Building the corpus data is time consuming task. So, only a limited number of speech data base is built for the selected part of 28 Ge'ez characters. This would limit the number of words and its application to be isolated word recognition instead of continuous speech recognition.

There are a number of speech recognition studies for Amharic language .when we come to Tigrigna language there are studies related to TTS. Tesfaye [38] is attempt made to develop a prototype TTS system for the Tigrigna language it is based on the concatenation of diphones using TD-PSOLA (time-domain pitch synchronous over lap and Add) technique. He used his voice to record an inventory of speech from which diphone units were extracted. The Tigrigna text to speech system has two major distinct

parts, which are text processing followed by speech synthesis. Visual c ++ programming language is used to develop an interface and to handle text processing and MATLAB programming language is used to handle the signal processing. Two major activities were made while preparing the diphone data base. Careful selection of corpus words and extraction of diphone from these words .To record corpus words and then to extract diphone from these words the free soft ware called *praat* is used. For testing the system the mean opinion score(MOS) testing method was adopted and the average result computed was found to be 3.05 which is closed to scale level good i.e. the system is a good start to producing realistic speech from text ,but there are several areas that can be improved. Inclusion of acronym converts to text processing modules and prosody control are some of the thing that need further research.

However to my knowledge no attempt is made to develop Tigrigna speech recognition, hence, this research is initiated to investigate the possibility of designing Hidden Markov model based speaker independent continuous Tigrigna speech recognizer.

CHAPTER FOUR

TECHNIQUES FOR DESIGNING TIGRIGNA SPEECH RECOGNIZER

This chapter describes techniques, tools and modules, language model and basic principles of HMMs, including the three basic algorithms and how to choose parameters for HMMs. Additional components such as decision tree clustering which is used for model refinement need during designing the prototype are included and discussed.

4.1 Language model

A language model is basic in designing speech recognition system .In studies a back-off system is used for modeling Tigrigna language. A back-off system has three main components: the back-off node that connects the unseen and less seen bi-grams, a back-off weight and a unigram probability P(wj) for each word. Unigram probabilities have to be smoothed, too, because unigram probability distributions may not be in balance as in the case of bi-grams. In this thesis, back-off bi-gram models are built using the given formulae below [21].

L: Number of distinct words; N (i,j): number of occurrences of the word pair (w_i , w_j)

N (i): number of occurrences of the word w_i. ;For unigram probabilities, P (i);

$$p(i) = \begin{cases} -N(i)\backslash N, & if\ N(i) > u \\ u\backslash N, & otherwise \end{cases} \quad \ldots\ldots \quad 4.1$$

Where u is unigram floor count set by the user and

$$N = \sum_{i=1}^{L} \max[N(i), u] \,. \quad \ldots\ldots..4.2$$

The back-off bigram probabilities are given as;

$$P(i, j) = \begin{cases} [N(i, j) - D] / N(i) & \text{if N(i, j)} > \text{t} \\ \alpha(\text{i})\text{P(j)} & \text{otherwise} \end{cases} \quad \ldots\ldots\ldots 4.3$$

Where D is a discount and t is bigram count thresholds that are set by the user. The back-off weight α(i) is computed to ensure that:

$$\sum_{j=1}^{L} P(i, j) = 1 \,. \quad \ldots\ldots\ldots\ldots 4.4$$

It is worth of note here that the back-off probability strictly depends on the quantity P(j). It performs a smoothing over the observed bi-gram probabilities. But we think that it would be better to associate a back-off weight α(i) that takes into account the word pair (w i , wj); thus it could be denoted as α(i,j).

4.2 Hidden Markov Model

A hidden Markov model is not used only in acoustic modeling. It has a wide application area in statistical signal processing. The basic component of a Markov model is the state. These states correspond to the positions of the vocal apparatus of the human. These states are hidden and the underlying task in an ASR is to decode the actual state sequence that causes the speech signal be observed. Then, the word (words) or phoneme (phonemes) to which these states belong is determined via the help of the language model. HMM have various elements [41, 15].

1. There are a finite number of states in the model, N. Within a state the signal has some measurable and distinctive properties.
2. At each time instance, t, a new state is entered based on a transition probability distribution that depends on the previous state.
3. After each transition is made, an observation symbol is output according to probability distribution which is related to the current state. This probability distribution is fixed for each state.
4. The initial state distribution that determines the state in which the system can be at the start up.

The definitions used in formulas are as follows:

$S= \{S_1, S_2 \ldots S_N\}$, the individual states

q_t: the state entered at time t N: the number of states

$a_{ij} = P[q_{t+1} = S_j \setminus q_t = S_i]$, the transition probability that the state entered at time t+1 is S_j given that the state, q_t, at time t was S_i. $1 \leq i,j \leq N$

$A= \{a_{ij}\}$, the state transition probability distribution. A is called as transition probability matrix

o_t : observation vector at time t

$b_j(o_t) = P[o_t \setminus q_t = S_j]$, probability of observing o_t given that the system is in state

S_j at time t , $1 \leq j \leq N$

$\pi_i = P[q_1 = S_i]$, probability of being in state i at time 1.

$\pi = \{\pi_i\}$, the initial state distribution. $1 \leq i \leq N$

$\lambda = (A, B, \pi)$, the compact notation to indicate the complete parameter set of HMM.

4.2.1 HMM Assumptions

There are a set of assumptions implicit in the structure of hidden Markov models about the structure of the process that they represent, though all the assumptions are not necessarily be correct [24].

The first assumption states that the observation accurately represents the speech signal. Normally the observation takes the form of some type of short term (like 10ms) spectra. These will not exactly represent the underlying speech. However it can be said that the speech is nearly stationary over such short periods and the observation is a reasonably accurate representation of the speech. This assumption is important in that prohibitively large amount of data would have to be processed otherwise [18].

The second assumption dictates that the likelihood of generating each observation is dependent only upon the state and is independent of all the other observations. This is not typically true of speech since the spectrum tends to change only slowly compared to the frame rate to ensure that the parameterized observations are an accurate representation of the original signal. However augmenting the observations with derivative parameters can reduce the effect of the resulting high degree of correlation between subsequent observations. When these are used the correlation does not seem to have an adverse effect on the recognition and attempts to model the correlation explicitly have not succeeded yet. The third assumption underscores that the state transition probabilities remain unchanged.

4.2.3 HMM topologies for speech recognition

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

The second phase the sound of the phone is assumed to be pure and in the third phase the sound is released and the vocal tract starts to transit to the next phone. This is called the off-glide, some overlap with the next phone may occur here. This suggests that at least three states should be used in a phoneme based HMM [41].

The three states, the first could represent the transition in to the phoneme, the second bears the steady state portion and the third depicts the transition out of the phoneme [17]. Adding more states means introducing more parameters and thus more degrees of freedom. Variations in a phoneme can be modeled more accurate but this also introduces a need for more training data to avoid under training. But a model should not be too large, a five state model does not work for phones that only occupy three time frames, so in larger models there should always be a 'short-cut' that can handle the shortest example in the training data [38]. These concepts can be seen on Figure 4.1 the topology shown in Figure is not the only choice to be built. It can be modified according to the application.

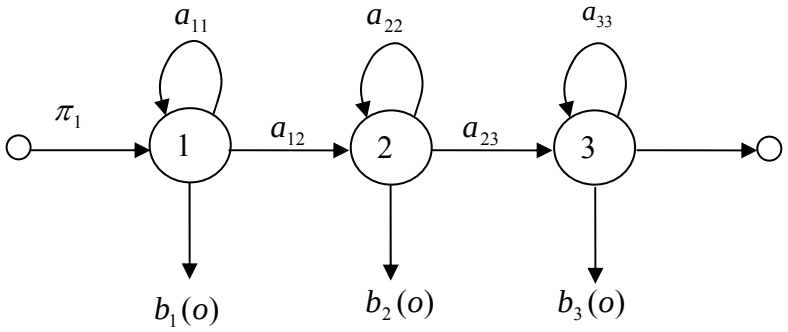


Figure 4.1 Topology for speech recognition

4.2.4 The three HMM problems

For an HMM to be useful in building speech recognizers, there are three basic problems of interest that must be solved for the model to be useful in real-world applications. These problems are the following: [27]

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

This problem is a problem of evaluating how well a given model matches a given observation [27]. To solve this problem the forward-backward algorithm is used.

Forward-Backward Algorithm is an algorithm that consider the forward variable $\alpha_t(i)$ defined as:

$$\alpha_t(i) = P(o_1\ o_2 \ldots\ldots o_t\ ,\ q_t = S_i \backslash \lambda) \quad \ldots\ldots\ldots\ldots.4.5$$

i.e., the probability of the partial observation sequence O1 O2 ……. . O_t, (until time t) and state S_i at time t, given the model λ. We can solve for α_t (i) inductively, as follows:

1. Initialization

   $\alpha_1(i) = \pi_i b_i(o_1) \qquad 1 \le i \le N$ …………………………….. 4.6
2. Induction

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(o_{t+1}) \qquad 1 \le t \le T\text{-}1\ ,\ 1 \le j \le N \quad \ldots\ldots.. 4.7$$

3. Termination

$$P(O \backslash \lambda) = \sum_{i=1}^{N} \alpha_T(i) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots.4.8$$

Termination stage is just a sum of all the values of the probability function α_t (i) over all states at instant T. The result represents how likely the given model produces the given observations.

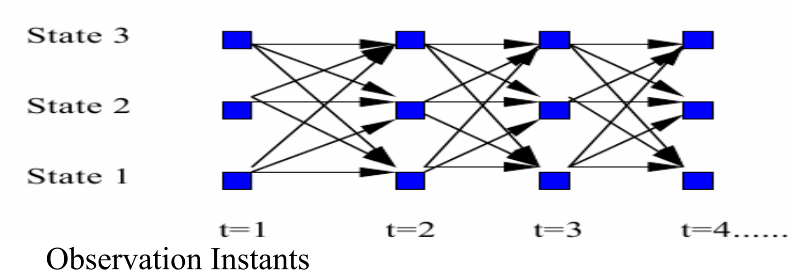


Figure 4.2 the flow of the process of a HMM in time.

The backward procedure is similar to the forward procedure. However, the state flow in computation is backward from instant T to instant 1. Let us define a back ward probability function βt(i);

$$\beta_t(i) = P(o_{t+1}, o_{t+2} \ldots\ldots o_T \setminus q_t = S_i, \lambda) \quad \ldots\ldots\ldots\ldots\ldots 4.9$$

1. Initialization

$$\beta_T(i)=1 \qquad 1 \le i \le N \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots 4.10$$

These initial values for β's of all states at instant T can be arbitrarily selected.

2. Induction

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}) \beta_{t+1}(j) \qquad t = T\text{-}1, T\text{-}2, \ldots 1 \; ; \; 1 \le i \le N \quad \ldots\ldots\ldots 4.11$$

The probability P(O\λ) can be computed from both forward and backward functions. This is illustrated in Figure4.4. The forward probability is a joint probability whereas the backward probability is a conditional probability. The probability of state occupation is determined by taking the product of the two probabilities.

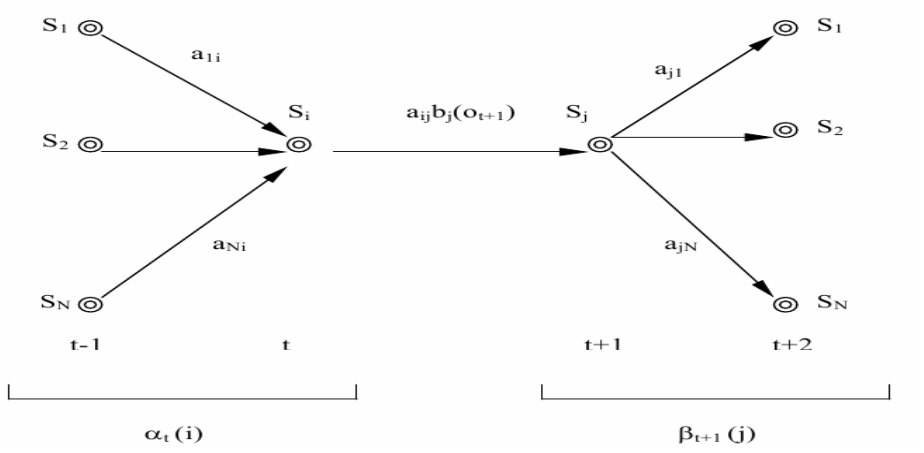


Figure 4.3: Forward-backward probability functions to obtain P(O\|λ)

Problem 2: Given the observation sequence O = { O_1 O_2 O_3 ..... O_T} and the model M, how do we choose a corresponding state sequence Q = q1, q2, ….qT which is optimal in some meaning full sense? Viterbi algorithm is used to solve this problem.

Viterbi Algorithm is find the single best state sequence, Q = {q_1 q_2 q_3 ..... q_T}, for the given observation sequence O = { O_1 O_2 O_3 ..... O_T}, we need to define the quantity

$$\delta_t(i) = \max_{q_1, q_2 \ldots\ldots, q_{t-1}} P(q_1, q_2, \ldots\ldots, q_t = i, O_1 O_2 \ldots O_t \mid \lambda) \quad \ldots\ldots.4.12$$

i.e., $\delta_t(i)$ is the best score (highest probability) along a single path, at time t, which accounts for the first t observations and ends in state S_i. By induction we have

$$\delta_{t+1}(j) = \left[\max_i \delta_t(i).a_{ij}\right] b_j(O_{t+1}) \quad \ldots\ldots\ldots. 4.13$$

To actually retrieve the state sequence, we need to keep track of the argument which maximized 4.7, for each t and j. We do this via the array $\psi_t(j)$. The complete procedure for finding the best state sequence can now be stated as follows:

1. Initializing

$$\delta_1(i) = \pi_i \, b_i(o_i) \qquad 1 \le i \le N$$

$$\psi_1(i) = 0$$

2. Recursion

$$\delta_t(j) = \max_{1 \le i \le N}[\delta_{t-1}(i) a_{ij}] b_j(o_t) \qquad 2 \le t \le T \;,\; 1 \le j \le N$$

$$\psi_t(j) = \arg\max_{1 \le i \le N}[\delta_{t-1}(i) a_{ij}] \qquad 2 \le t \le T \;,\; 1 \le j \le N$$

3. Termination

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

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

4. Backtracking

$$q_t^* = \psi_{t+1}[q^*_{t+1}] \qquad T\text{-}1 \ge t \ge 1$$

It should be noted that the Viterbi algorithm is similar (except for the backtracking step) in implementation to forward calculation of solution to problem 1.

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

Baum-Welch Algorithm is related to the training problem that is the most difficult one. The aim is to adjust parameters of the model according to an optimality criterion Baum-Welch algorithm is strictly related to forward-backward algorithm and it tries to reach the

local maxima of the probability function P (O\λ). The model always converges but the global maximization is not guaranteed. That is why the initial point of search is very important. Let us define the probability of being in state S_i at time t, and state S j at time t+1;

$\xi_t(i,j) = P(q_t = S_i , q_{t+1} = S_j \backslash O,\lambda)$

Its relation with forward and backward variables is;

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

The numerator term equals to P ($q_t = S_i , q_{t+1} = S_j \backslash O, \lambda$) and the denominator term equals to P(O\λ). Now define the posteriori probability of being in state S_i at time t, given the observation sequence and the model;

$$\gamma_t(i) = P(q_t = S_i \backslash O,\lambda)$$

Its relation with forward and backward variables is;

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

If we sum γ_t (i) over the index t, we get a quantity which can be interpreted as the expected number of times that state S_i is visited and the summation of ξ_t (i,j) over t can be interpreted as the expected number of transitions made from S_i to S_j. With the help of these, the formulas for re-estimating the parameters A, B and π are given as;

$\bar{\pi}_i$ =expected number of times in state S_i at time (t=1) = $\gamma_1(i)$

$$\bar{a}_{ij} = \frac{\text{xpected number of transitions from } S_i \text{ to } S_j}{\text{expected number of transitions from } S_i}$$

$$= \frac{\sum_{t=1}^{T-1}\xi_t(i,j)}{\sum_{t=1}^{T-1}\gamma_t(i)}$$

Expected number of times in state j and observing symbol v_k expected number of times in state j

$$= \frac{\sum_{\substack{t=1 \\ O_t = v_k}}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

After the re-estimation of the model parameters we will have a new model which is more likely to produce the observation sequence O. The iterative re-estimation procedure continues until no improvement in P (O\λ) is achieved.

4.3 Model Refinement

One has to build more accurate models, whereas he is obliged to obtain the parameters from sparse training data. Accuracy of the model can mean using more parameters per model or increasing the number of sub word models. One way to increase the accuracy without increasing the complexity of the model is to build models that share parameters. This sharing is also called as tying. Parameter tying can be done in different levels in Gaussian mixture based ASR systems. Means, co-variances, mixture components, states or transition matrices can be tied.

State tying cannot be done arbitrarily. It can be achieved via data driven clustering or decision tree clustering. In both cases, the training data is used to determine which states should be tied. In data driven clustering, the most similar states are tied. Similarity measure is the distance between these states. In Decision tree clustering (DTC), optimum decision tree is found and the states remaining in the same leaf of this tree are tied. In this thesis, DTC is used. That is why DTC will be given here.

4.3.1 Decision tree Clustering

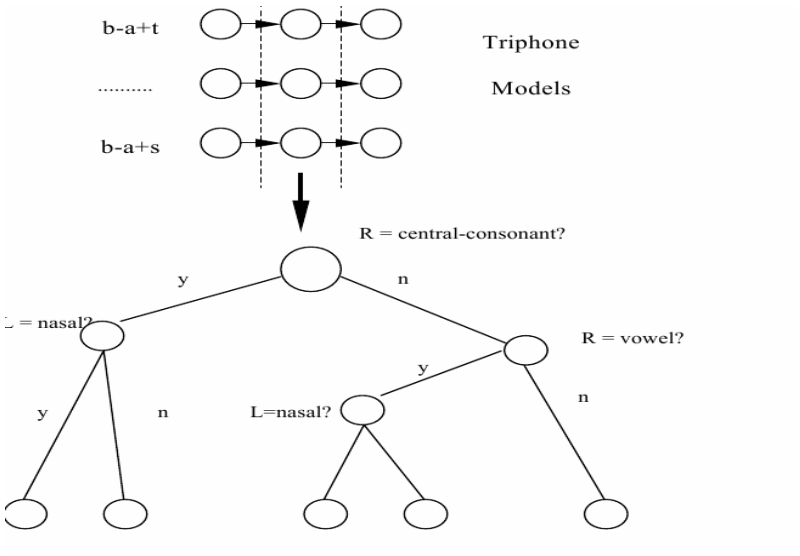


Figure 4.4 clustering the center states of the phone ‘a’

One major property of decision trees is the capability to produce models that are not seen during acoustic training. Decision trees constructed are stored to use in the future. Referring to these trees, the unseen models can be constructed artificially.

Decision trees are built by asking a set of binary questions, i.e. the answer is yes or no, when clustering the states. These questions are asked at branching nodes of the tree. Initially, all states belonging to models to be clustered are placed in the root node of the tree. As the questions are being asked, the space of states is divided into two. The number of states in a half space depends on the answer ‘yes’ or ‘no’. Splitting the state space ends up when the optimality criterion is met. This criterion depends on two different thresholds defined by the user. The threshold can be the minimum number of states that must be in one cluster or the increase in the total likelihood of the training data on the current set of clusters. Then the states in the cluster are tied.

An example of decision tree built for triphone models can be seen in Figure 4.4 [28]. The questions used in this example are like; “Is the right context a central-consonant?”, or “Is the left context a nasal?”. These questions for a triphone based system can be denoted as;

QS R-nasal {*^m+*,*^n+*,*^N+*}

QS R-nasal {*^m+*,*^n+*,*^N+*}

The questions used in thesis can be found on Appendix F. This question list is mainly taken from [12], but some modifications are made according to the phonemes used in this thesis. Clustering is ends up not according to the number of questions, but according to the optimality criterion.

4.4 Tools and Modules

The description of a software package in a research monograph is unusual. However, the use of HTK (Hidden Markov Model Toolkit) has been a fundamental part of speech recognition research and at the same time, resolving certain issues is not so trivial. The description here has been prepared to provide a better understanding of what has been done in this thesis and of the process involved in the recognition task by utilizing the modular structure of HTK.

The HTK is a software toolkit for building speech recognition systems. It can perform either continuous density, semi-continuous density or discrete probability HMM based tasks. It is developed by the Cambridge University Speech Group. It has been improved and added properties over years since the beginning of the nineties.
In this thesis, Version 3.4.1 is used to implement the recognition system. It is a freeware that can be downloaded from internet, but not with all of the features developed.

The HTK is designed to be flexible enough to support both research and development of HMM systems. By controlling the tool software via commands with some options as desired, a speech recognition system can be implemented, tested and then its results can be inspected. A wide range of tasks can be performed, including speaker independent or continuous speech recognition using models based on whole word or sub-word units. HTK consists of a number of tools that perform tasks such as coding data, various styles of HMM training including embedded Baum-Welch re-estimation, Viterbi decoding, producing N-best lists or single recognition result. It can perform results analysis and editing HMM definitions, too. Especially editing the HMM's externally enables the user to control the acoustic models with a considerable flexibility.

The HTK also supports language model constructing. The language model is based on n-grams. The version used is restricted to bigrams. General aspects of HTK are given in the following sections. The technical details and instructions how to use HTK can be found in HTK book for Version 3.4.1

There are four main phases when constructing a speech recognition system: data preparation, training, testing and analysis. The commands to use HTK and auxiliary software modules are related to these main phases. They have an abbreviated form of typing. These forms represent the commands executed in operating system environment, while some command-like abbreviations represent the module programs that are used by these commands. The commands (tools) use the modules when performing a user defined task. In this respect, the modules are internally embedded. However, the user can control these modules either defining an option in the command line or defining the desired parameters in a text file. Basically, HTK deals with two types of files: text files and speech data files. Text files can hold some editing commands, configuration parameters, transcription of speech files or the list of the files that will be used when performing the task. The processing stages are demonstrated in Figure 4.5.In the figure, the abbreviations in boxes are commands used in HTK.

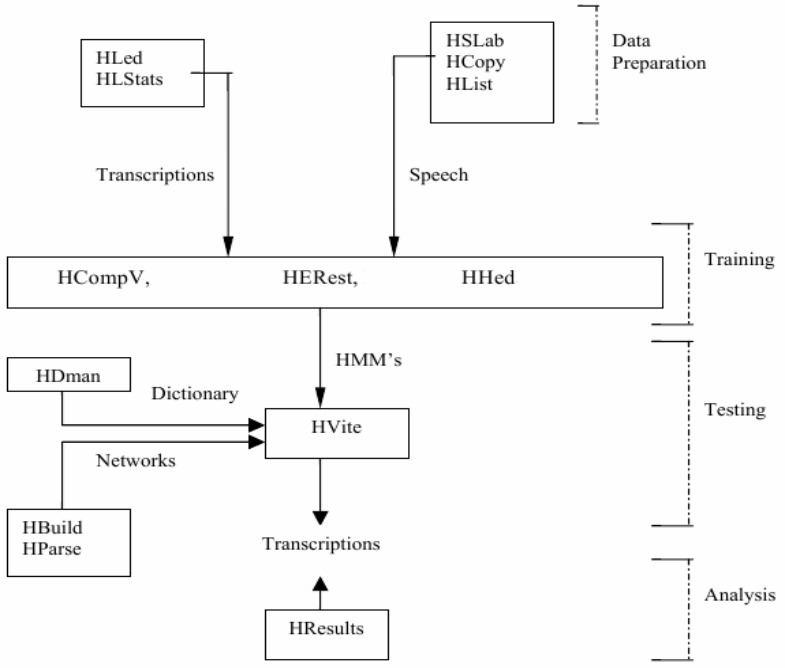


Figure 4.5 HTK processing stage

4.4.1. Data Preparation Tools

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

Although all HTK tools can parameterize waveforms on-the-fly, in practice it is usually better to parameterize the data just once. Thus the tool HCopy is used for this purpose of copying one or more source files to an output file. By setting the appropriate configuration variables, all input files can be converted to parametric form, as they are read-in. Therefore, simply copying each file in this manner performs the required encoding. HList can be used to check the contents of any speech file and since it can also convert input on-the-fly, it can be used to check the results of any conversions before processing large quantities of data.

HLed is a script-driven label editor which is designed to make transformations to label files, like translating word level label files to phone level label files, merging labels or creating triphone labels. HLed can also output files to a single Master Label File MLF, which is usually more convenient for subsequent processing.

HLStats can gather and display statistics on label files and where required. Using this tool in the end the transcription and the parameterized data files will be ready for training.

4.4.2 Training Tools

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

The purpose of the prototype definition is only to specify the overall characteristics and topology of the HMM. The training tools will compute the actual parameters later.
Sensible values for the transition probabilities must be given but the training process is very insensitive to these. An acceptable and simple strategy for choosing these probabilities is to make all of the transitions out of any state equally likely. If segmented transcriptions are available the tools HCompV provide isolated word style training using the fully labeled data as bootstrap data. HCompV can be used to provide initial estimates of whole word models in which case the observation sequences are realizations of the corresponding vocabulary word.

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

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

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

4.4.3 Recognition or Testing Tools

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

HVite can support cross-word triphones and it can run with multiple tokens to generate lattices containing multiple hypotheses. It can also be configured to rescore lattices and perform forced alignments.

The word networks needed to drive HVite are stored using the HTK standard lattice format (SLF). This is a text-based format and hence word networks can be created directly using a text-editor. However, this is rather tedious and hence HTK provides two tools to assist in creating word networks: Hbuild and Higher level language (EBNF). Hbuild allows sub-networks to be created and used within higher-level networks. Hence, although the same low-level notation is used, much duplication is avoided. Besides, HBuild can be used to generate word loops and it can also read in a backed-off bigram language model and modify the word loop transitions to incorporate the bigram probabilities. As an alternative to specifying a word network directly, a higher level grammar notation can be used. This notation is based on the Extended Backus Naur Form (EBNF) used in compiler specification the tool HParse is supplied to convert this notation into the equivalent word network. Whichever method is chosen to generate a word network, it is useful to be able to see examples of the language that it defines. The tool HSGen is provided to do this. It takes as input a network and then randomly traverses the

network outputting word strings. These strings can then be inspected to ensure that they correspond to what is required.

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

4.3.4 Analysis Tools

Hresult is one HTK tool that compares recognition results with original transcriptions. It uses dynamic programming to align the two transcriptions and then counts substitution, deletion and insertion errors. HResults can also provide speaker-by-speaker breakdowns, confusion matrices and time-aligned transcriptions. Using tools and techniques discussed in this chapter an attempt is made to design a prototype speech recognizer for Tigrigna language

CHAPTER FIVE

EXPERIMENTATION AND PERFORMANCE ANALYSIS

This chapter describes the design and construction of the speech recognizer for Tigrigna language that is capable of recognizing Tigrigna speech (sounds). The intention is to develop a system that is as general as possible, so that it can be used, with slight modifications and with some adaptive training, as a speech interface for many other different applications. To meet these requirements speaker independent continuous speech recognition is designed for Tigrigna language and implemented using phonemes as base unit. Thus, the system's vocabulary was not limited in any way to some fixed set of words. If a new vocabulary is required to be incorporated, what needs to be done is only modifying the language model and adding the new word to the pronunciation dictionary.

The development process is performed on the Lunix fedora 8.0 platforms using the tools in HTK. Various preprocessing programs and scripts are also used as shown in the Figure 4.5. The discussion of the experiment is presented in accordance to the steps that should be followed while building such speech recognizer. Accordingly, the chapter is organized in to four parts; preprocessing, Training, Optimization (Enhancement) and Evaluation.
The preprocessing part presents the way the data preparation task was performed, the developed language model and the prepared dictionary. The training portion deliberates on how the acoustic models were developed, the steps followed in training and building the monophone level recognizer, and the tools with their accompanying scripts used in the process. It also discusses the mechanisms followed in refining and optimizing the monophone recognizer. The final part is on the testing and performance evaluation of the systems at various stages.

5.1 System design

Systems usually have a number of components that interact to each other in a logical fashion. The processing logic of a component may generate an output that may be the

input to another component, may also trigger another component to activate itself or may affect the processing logic of another component. Hence, in such cases it will be very important to clearly put the system design that shows the whole process of the system and the interaction among components.

Speech Recognition systems incorporate two basic components: the Natural Language Processing (NLP) and Digital Signal Processing (DSP). The natural language processing deals with the pre-processing activities includes speech analysis and language modeling (phonetic). Digital Signal Processing is responsible for text generation from speech input.

Speech recognition for different languages mainly differs because of the processing logic of the NLP component. Multiple languages are different phonetically, morphologically, syntactically, and semantically that leads to different ways of doing the natural language processing component of the speech recognition. However the processing logic of the speech recognition for Tigrigna using HMM is similar to other language's that uses HMM technique with regard to DSP. However, there is a difference as far as the NLP component is considered. Hence, the HMM based speech recognition for Tigrigna will address the speech (sounds), and phonetic analysis deal with issues related to language's characteristics.

Figure 5.1 shows the system design of the Tigrigna speech recognition using HMM. As the system design shows, the recognizer has two parts: The training and recognizer part. The training part includes data preparation, language modeling, feature extraction, and building the HMM. Whereas, the recognizer part includes preparing labeled speech from the speech input, selecting appropriate HMMs, extracting speech parameters from HMMs, and finally generating the speech waveform from the speech parameters.

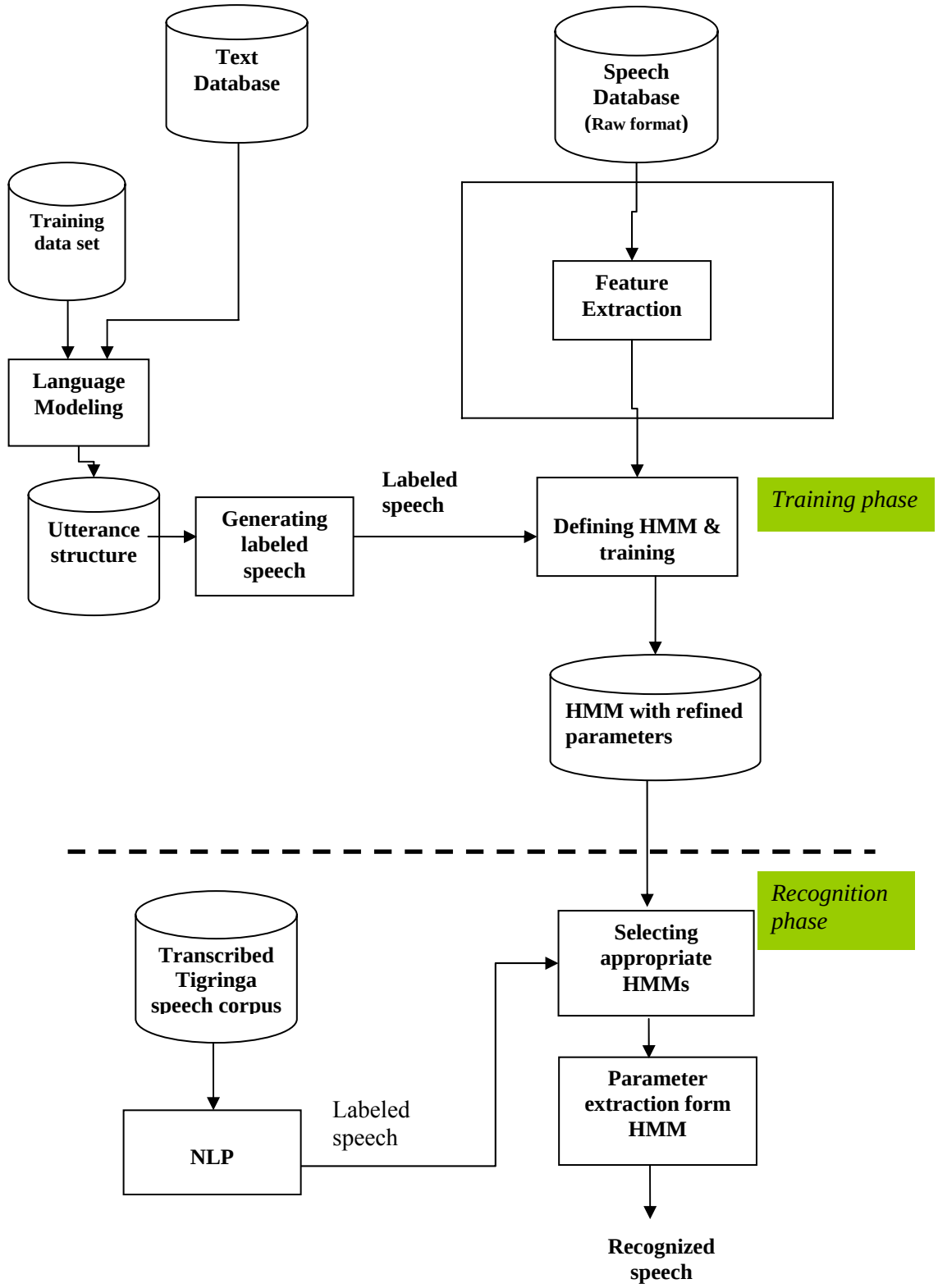


Figure 5.1 Overview of the system design of Tigrigna Language using HMMs

5.2 Language Model

Out of the total system design, the language modeling component is responsible in adding phonetic and acoustic information into the speech that will be used for training the model. As it is shown in Figure 5.2, this component includes activities like, transcription of the original speech, labeling the utterance, and generating the utterance structure. In the next subsections each activity of the language modeling are discussed in detail.

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

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

Using the word level transcriptions and statistics on the number of occurrences of each word and each combination of two words, the probability matrix was prepared. These statistics were then used to create backed-off bigram language models for the training, test and evaluation sets, using the HBuild tool which translated the gathered statistics into HTK Standard Lattice Format, that are used for storing word models and multiple hypotheses from the output of a speech recognizer.

In language modeling, the problem is to predict the next word given the previous words It is fundamental not only to ASR but also to statistical machine translation (SMT), question answering, text summarization, spelling checkers, information retrieval, etc., [49]. Statistical language models are the most commonly used types of language models [10]. Since different languages have different structure, language models are also basic language specific resources so that can handle all.

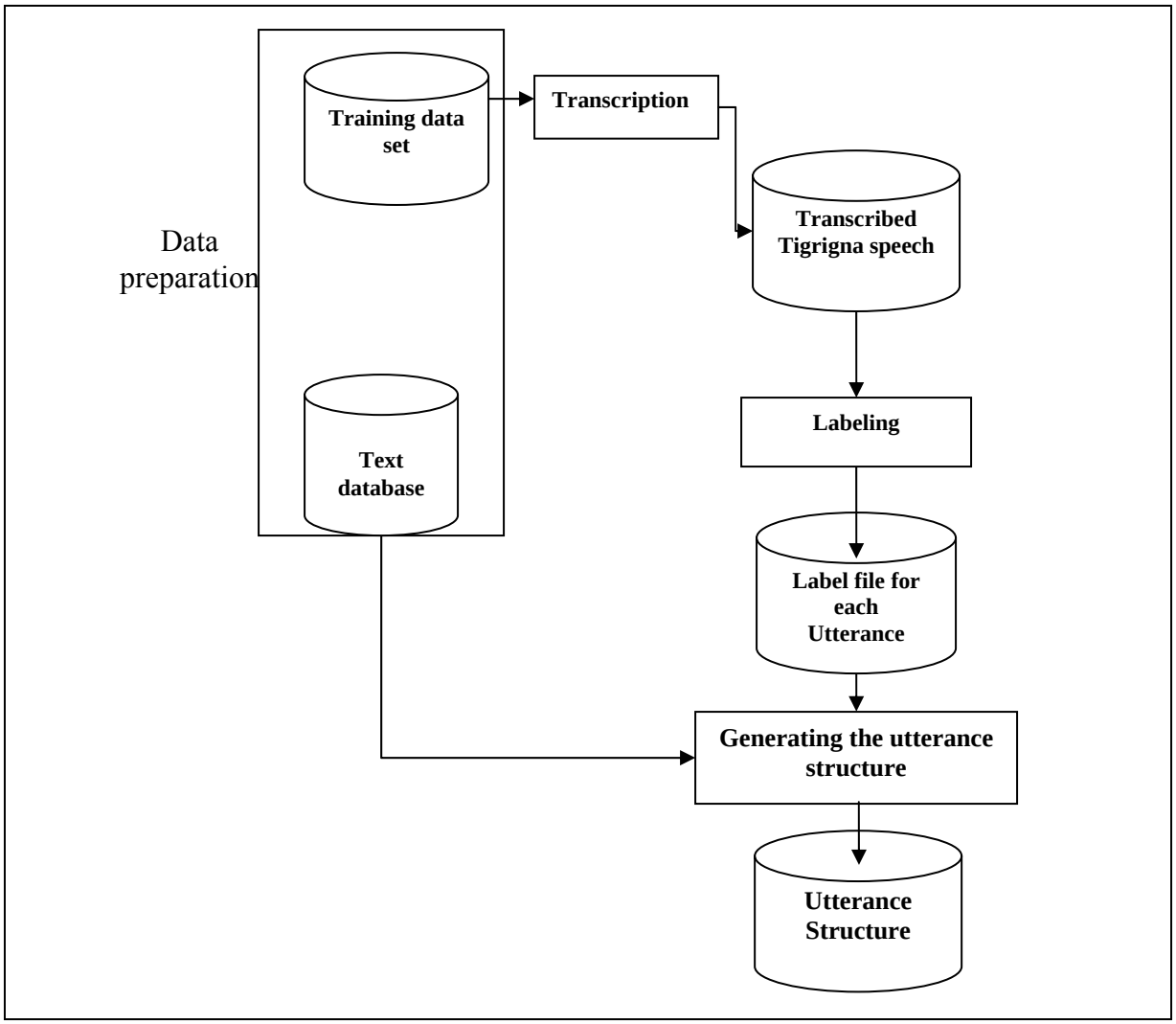


Figure 5.2: Language modeling system design

5.3 Data Collection and Preparation

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

we describe our efforts in building up speech data, text data and lexical dictionary for Tigrigna speech recognition.

5.3.1 Text Data

To create more general acoustic models sentences are selected from one fiction books namely “Hezike nabey ” and one news paper namely “Weyn” .This fiction book and news paper are selected because phonetically rich and balanced collection of sentences and HMM based speech recognition technique requires each phoneme to be included in the training dataset so many times, with different contexts, so as to model each phoneme as accurate as possible. During the process of text selections some corrections have done such as spelling and grammar errors has been corrected, long sentences are shortened and numbers have been textually transcribed . The selected text contains 300 sentences. These selected sentences have been manually checked for grammatical, spelling, and abbreviations and have been manually corrected.

5.3.2 Speech Data

It is an already established fact that in order to build a robust large vocabulary speech recognizer, all the acoustic (sub-word) models should receive enough training examples. In due course, the many variations that can occur in speech, due to, for example, variation in speed of speech, type of speaker, and gender. It follows that a training corpus should be sufficiently large, consisting of speech samples from wide range of age group 20-60. Thus, as mentioned in the methodology section, the data base used for this work is a corpus from 12 persons and contained 300 sentences spoken by 12 persons, each having read 25 utterances. The utterances are also comprised of all Tigrigna phonemes in many phonetic contexts. The database, however, is prepared from utterances made by people whose first language is Tigrigna. The data base is divided into training dataset and test dataset, with ratio of 5:1 as shown in tables 5.1 .The train dataset is used for tuning during development of the system and the test dataset for evolution performance of the system.

In general after selecting the text, the next step in preparation of Tigrigna speech corpus is recording speech. In the recording process of the selected sentences the speaker asked to read exactly what is presented to. The sentences are given for the speaker in a printed format. At the process of recording degree of control was very high, each utterance of the speaker is checked directly for errors and if an error is found, the speaker is asked to re-read the text again and again. The text has been read in an office environment.

	Train dataset	Test dataset
For speaker independent	250 sentences	50 sentences
For speaker dependent	100 sentences	20 sentences

Table 5.1 the dataset used in this experiment

5.3.3 Dictionary preparation

A pronunciation dictionary is a machine-readable transcription of words in terms of sub-word units. It specifies the finite set of words that may be output by the speech recognizer and gives, at least, one pronunciation for each. The pronunciation dictionary, i.e. the lexical model, is one of the most important blocks in the development of large vocabulary speaker independent recognition systems. It describes the sequence of HMMs that constitute each word. Pronunciation dictionary is a valuable resource in the process of building a sub-word based recognizer. However, if it is produced manually, it can require considerable investment. For English and other technologically favored languages, commercial and public domain dictionaries are available. This experiment is at a disadvantage in this respect and the pronunciation dictionary is constructed manually and letter-by-letter spelling of each word is used as its pronunciation. Since sub-word units are needed to build the models, a dictionary based on phoneme is required.

The HTK lexical dictionary format is used to create lexical for Tigrigan language.
In order to build a dictionary the first thing to do is create a sorted list of the required words. To build robust acoustic model, it is necessary to train them on a large set of sentences containing many words. The word lists are extracted from the data which we

have found from variation patterns. Before using HTK toolkit, one would need to edit the text into a suitable format. For example, it would be necessary to change all white space to new lines and then to sort the words into a unique alphabetically ordered set, with one word per line. Some portions of this pronunciation dictionary are attached as Appendix C. The dictionary for use in HTK has a very simply format. Each line consists of a single word and the corresponding pronunciation. The general format of each dictionary entry is:

WORD [output] p1, p2, p3 . . .

which means that the word '*WORD'* is pronounced as the sequence of sub-word units p1,p2,p3,… and each sub-word unit is represented by an HMM. These sequences of models are used in recognizing the words. The string in square brackets specifies the string to output when that word is recognized. If it is omitted then the word itself is output. If it is included but empty, then nothing is output as in the case of '*SILENCE'* where the entry is:

SILENCE [] sil to mean the entry '*SILENCE* 'has *'sil'* as its pronunciation and null output symbol, then, using the tool '*HDman'* in HTK and the produced dictionary, a new dictionary was created with a short pause at the end of every pronunciation .'*HDman'* takes a name of the new dictionary to be created and the original dictionary as its parameters plus other optional parameters. The following command was executed to generate the dictionaries, in batch:

HDMan -m -w *wlist* -n monophones1.-l dlog1 *dict wlistronu*

'*Monophones1'* which are preceded by the –n format specifier, contained lists of all the unique monophone sounds in the database and are generated by the tool. At the end of both the phoneme lists, '*sil*' is added manually to model the longer silence.

The global.ded script was used to attach '*sp*' at the end of each entry of the dictionary: The following command is used form:

AS sp

MP sil sil sp

The AS command append the' *sp'* model at the end of each pronunciation. The model is automatically appended at the end of every word out of a reasonable assumption that there would usually be a very brief pause between words. If the dictionaries contain any silence markers then the 'MP' command will merge the' *sil* 'and '*sp'* phones into a single '*sil'*. However, the pronunciation dictionary is not handles the problems of gemination of consonants and irregular realization of the sixth order vowel and the glottal stop consonant. These problems have a direct effect on the quality of sub-word transcriptions.

5.3.4 The Task Grammar or a Word Network

A word network describes the sequence of words that can be recognized by the recognizer. A word level network will typically represent either a Task Grammar which defines all of the legal word sequences explicitly or a word loop which simply puts all words of the vocabulary in a loop and therefore allows any word to follow any other word. In this experiment, it is assumed that words are independent and equally probable; there is no grammar, whatsoever, that decides word sequences.

A word network is defined using HTK Standard Lattice Format (SLF). Standard Latice Format (SLF) is a low level notation of word networks in HTK, in which each word instance and each word to word transition is listed explicitly. An SLF word network is just a text file and it can be written directly with a text editor or a tool can be used to build it. In order to build the word network, the task grammar should be written first using a higher-level grammar notation. This notation is based on the Extended Backus Naur Form (EBNF) that is used in compiler specification. The tool HPARSE is supplied to convert this notation into the equivalent word network. The task grammar definition used in this experiment is attached as Appendix D. Once the task grammar is defined, a network is generated by the command:

HPARSE gram wdnet

Where' *gram* 'is the name of the file containing the above 'grammar' definition and '*wdnet'* is the file name for the SLF format network generated. In this file each word instance and each word-to-word transition is listed explicitly .A portion of the SLF file generated in this experiment is attached as Appendix E. The speech recognition task

completely defined by its network, dictionary, and HMM set is ready for use. The tool HSGen is used to be sure that no mistakes are made when writing the grammar.

```
HSgen -q -s -n X wdnet  dict
```

The –n option used to state the number of sentences, X, to be generated out of the model. -n sets the total number of sentence generated to be N (default value 100). However there are words like “menen” and “Aawdii” has been errors because of they have no word separation between the words, and then correcting by inserting “|” between the words.

5.3.5 The Phoneme Set

As was already mentioned, the basic speech unit in the system is phonemes and they have to be chosen. However, there could be found no such thing as a phoneme set for a language, unlike the situation for example with the characters in the alphabet [41]. Existing sets differ in the number of details they provide. But for Tigrigna, Girmay [6] identified 29 phonemes that are used in standard dictionaries and in this research work all the phonemes are considered.

However, the transcription of the phonemes in the corpus is primarily in a font known as Ethiop. The phonemic representations of the font are not suitable for the tools in HTK toolkit. Thus they are renamed and modified for convenience in later activities and as per the requirements of HTK. Furthermore two other models are added. The first, *sil,* models longer periods of silence. These silences occur between sentences or when a person is not speaking at all. The second phoneme, *sp*, also represents silence, but only periods of short duration. This is the kind of silence that occurs between words. The former represents periods of pure silence that can greatly vary in length. It usually demarcates sentence boundaries. While the later represents optional periods of silence shorter than the duration of a phoneme, and are usually found between words. Appendix B. depicts the phonemes used in their original notation, the notation in this experiment, is an exemplary word for each phone and phonetic transcription of the words.

5.4 Transcription Files

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

However, the examples in the training data set contain complete sentences, which are strings of phones. Thus to ensure that the right part of an utterance is used to train a model, a transcription of the utterances at phone level is needed. Besides, to evaluate the performance of the final system, textual transcriptions of what is really said in the training, test and evaluation utterances are needed. These phone level transcriptions are created from the word level transcriptions and then a file called Master Label File (MLF) containing all the transcriptions of the words in the utterances, indexed by the file name of the utterances is produced. All the utterances have unique file names and the file prepare in a label file format required by HTK. Then using pronunciation dictionary and the above phone level MLF, the phone level transcription is created by the tool *'HLed'* in HTK. It takes the word level MLF, the pronunciation dictionary and an editing script as its arguments. The '*HLed* 'edit script '*mkphones0.led*' contains the following commands:

EX
IS sil sil
DE sp

The '*EX* 'command is replaces each word in '*master.ml'* by the corresponding pronunciation in the dictionary file '*dict'*. The IS command inserts a silence model '*sil* 'at the start and end of every utterance. Finally, the delete '*DE* 'command deletes all short pause '*sp'* labels, which are not wanted in the transcription labels at this point.The following batch file runs containing two HLed commands:

HLed –l * -d *dict* –i *phones1.mlf* mkphones.led *words.mlf*

As a result a phone level transcription is prepare, one with '*sp* 'at the end of each word and the other with out.

5.5 Coding and feature vector extraction

The other important task in data preparation is the extraction of feature vectors i.e. parameter the raw speech waveforms into sequences of feature vectors. Most digital representation methods are based on the assumption that parameters of speech remain unchanged over a short-time period [28].Parametric representations may be divided into two groups[41]:

- Those based on the Fast Fourier Transform spectrum (FFT), and
- Those based on Linear Prediction Spectrum (LPC)

The commonly used short time spectrum can be obtained using the Fast Fourier Transform method. Though HTK supports both FFT-based and LPC-based analysis, Mel Frequency Cepstral Coefficients (MFCCs) which are derived from FFT-based log spectra, consisting of the length of the parameterized static vector (+13), the delta Coefficients (+13) and the acceleration coefficients (+13) which add up to 39 are used in this experiment. In order to code the data, a configuration file (config) is created which specifies all of the conversion parameters uses in figure 5.4 for training.

The settings in the configuration file specify that the target parameters are to be MFCC, A sampling rate of 10 ms (HTK uses units of 100ns), is used with an overlapping window of 25 ms the output should be saved in compressed format, a CRC checksum should be added, and so on. Coding can be performed using the tool '*HCopy'*. A script file containing the list of each source file and its corresponding output file is supplied to the '*HCopy,* Command as an argument. The first few lines of this script file look like:

s1.wav s1.mfc
s2.wav s2.mfc
s3.wav s3.mfc
(etc.)

Given that the above script is stored in the file' *train.scp'*, then the following command is executed to parameterize all the 250 training speech utterances:

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

A similar procedure is used to code the test data after which all of the pieces have been put in place to start training the HMMs.

5.6 Training the Models

Training the model means estimating the HMMs parameters, which are the mean, the variance, and the transition probabilities based on the utterance structure and the extracted parameters (features) [43]. Once the parameters are extracted, training of HMMs is performed with the Hidden Markov Model Toolkit, which is software that provides a set of library modules and tools for building and manipulating HMMs. HTK supplies basic tools for HMM parameter estimation: *HCompV* and *HERest* [43]. In addition to these tools one additional HTK tool is used for context analysis, which is *HHEd.*

HCompV is used for initialization of the model parameters. *HCompV is* used to set the mean and variance of every Gaussian component in a HMM definition. This tool is typically used for initializing the model which computes the parameters of a new HMM. In this research work *HCompV* is used to initialize the model since the speech utterance is labeled. HERest is used to refine the parameters of existing HMMs using Baum-Welch Re-estimation algorithm [43].

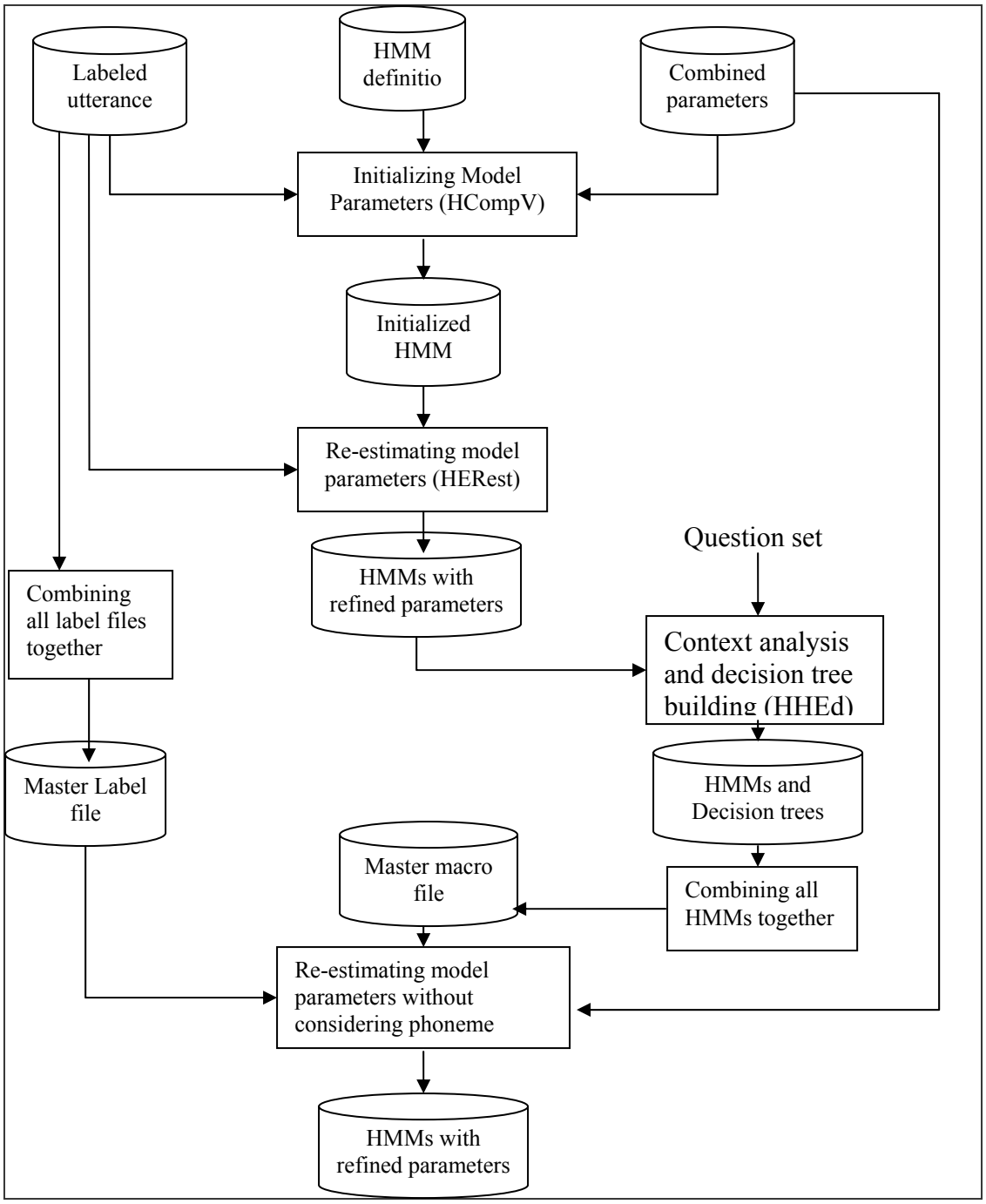


Figure 5.3: The procedures to generate HMMs for each phoneme.

In the following paragraphs each steps depicted in Figure 5.3 are described. As it is shown in Figure 5.3, the first step in generating HMM parameters is to provide initial estimates for the parameters of every HMM models of the phoneme set. This process

takes as input HMM definition, combined parameters, and label for each phoneme and initializes them one by one.

5.6.1 HMM Prototype

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

The topology of the model consists of the start and end states and three emitting states, using single Gaussian density functions. The states are connected in a left-to-right way, with no skip transitions. A five state topology choose because it resulted in better performance for Amharic language's sounds that are even longer than the Tigrigna phonemes. Single mixture is used [45].The prototype model adopt from the HTK handbook [43] is attached as Appendix F. Each vector was of length 39 which compute from the length of the parameterized static vector (MFCC_0_D_A) plus the delta coefficients (+13) and the acceleration coefficients (+13).In order to code the data, a configuration file (config1) is created which specifies all of the conversion parameters. The following is the setting used in this experiment.

# Coding parameters	USEHAMMING = T
TARGETKIND=MFCC-0-D-A	PREEMCOEF =0.97
TARGETRATE=100000.0	NUMCHANS=26
SAVECOMPRESSED=T	CEPLIFTER =22
SAVEWITHCRC=T	NUMCEPS =12
WINDOWSIZE =250000.0	ENORMALISE =F

Figure 5.4 coding parameters used for training.

5.6.2 Initial models

Initializing the HMM parameters requires some data (combined parameters). To avoid this problem, the speech signal of each phoneme will be uniformly segmented and each segment will be used to initialize the parameters of each sates of the model. Such initialization only makes sense if the HMM is left-to-right. In this work HCompV is used for model initialization process. Each operations of HCompV are given in Figure 5.5.

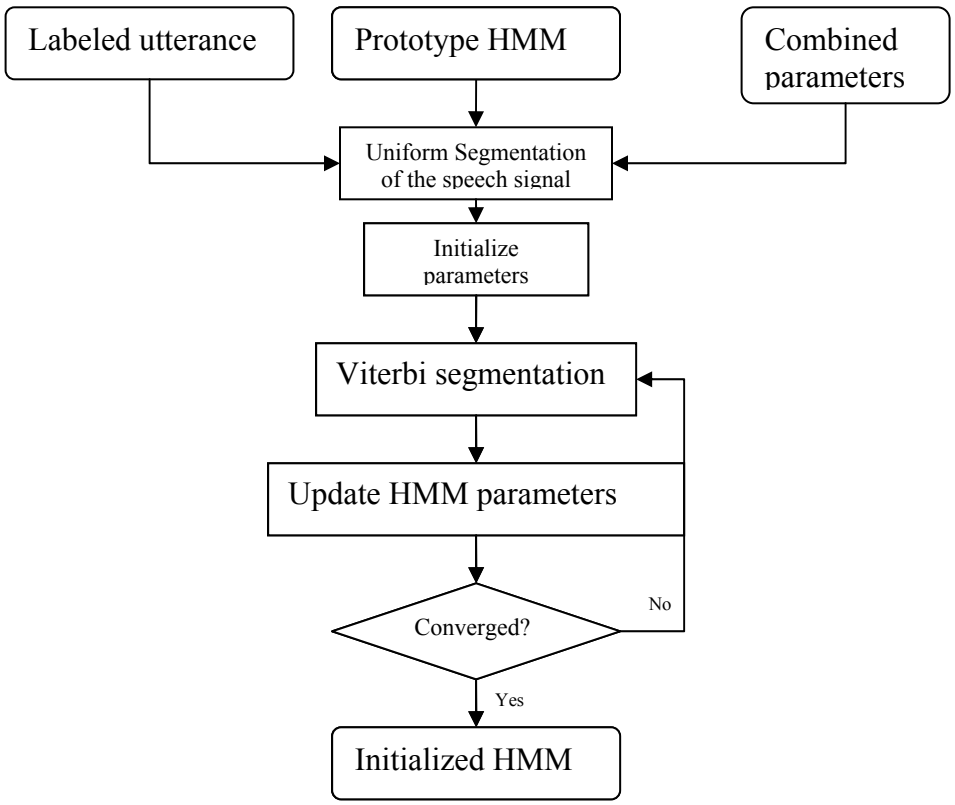


Figure 5.5: HCompV operation Adapted from [43]

Good initial models can be obtained by assuming an HMM as a generator of speech vectors. The training examples of the phones corresponding to the model whose parameters are to be estimated can be viewed as the output of this model. Thus if the state that generated each vector in the training data is known, then the unknown means and variances could be estimated by averaging all the vectors associated with each state [40]. In HTK this concept is implemented using the HCompv tool. Based on the above

prototype model, the tool HCompv is executed to create another average model.

Hcompv -A -C config1 -f 0.01 -m -S words.mlf -M hmm proto

This produced a new prototype version and stored it in the directory '*hmm*'. It scanned the set of training data files and compute the global mean and variance and set all of the Gausians in a given HMM to have the same mean and variance.The –f option is used to create a variance floor macro, called' *vfloor'*, which is equal to 0.01 times the global variance and is used to set the floor on the variance estimated in the subsequent steps.
Finally, a Master Macro File (MMF) which contained the initial model set by the name *hmmdefs* is created. This is done manually copying the prototype for each phone and renaming it accordingly.

5.6.3 Embedded re-estimation

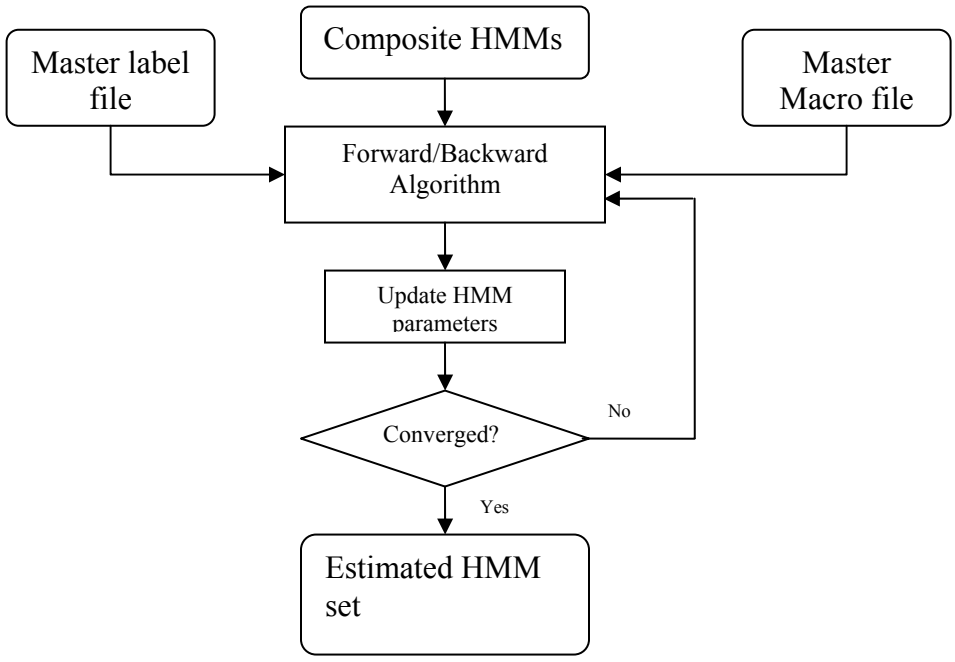


Figure 5.6: HERest operation Adapted from [43]

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

training all models are trained in parallel rather than individually. That is each utterance is processed in turn and the accompanying transcriptions are used to construct a composite HMM which spans the whole utterance. This composite HMM is made and implemented by concatenating instances of the phone HMMs corresponding to each label in the transcription. The Forward-Backward algorithm is then applied and the sums needed to form the weighted averages are accumulated. When all of the training files have been processed, the new parameter estimates are formed from the weighted sums and the updated HMM set is created. This embedded procedure is implemented in the HTK tool *HERest* . The operations of *HERest* are given in Figure 5.6. It was executed by writing the following command:

HERest -C config1 -I phones1.mlf -t 250.0 150.0 1500.0 -S words.mlf
H hmm0/macros -H hmm0/hmmdefs.mmf -M hmm1 monophones1.txt

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

5.6.4 Fixing the silence models

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

So far, training is done with out the short pause model *sp*, because this optional silence could be not assign to other models during initial trainings. But from this on, it is introduced using *monophones1*, the phone set with *sp,* which was generated using the

tool *HDman* and the phone level MLF, phones1. This MLF is made to contain *sp* at the end of each word while it is generated using *HLed;* out of a reasonable assumption that there will always be such unnoticed pause there in. or else, the transcription of each utterance had to be prepared in such a way that the *sp* transcribe whenever it is observed in all the recorded speeches. The model design to have three states, i.e., only one emitting state in the middle and two non-emitting states at left and right. Then the emitting state makes to be tied to the center state of the *sil* model resulting in the so called tee model.

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

HHed –H hmm4/macros –H hmm4/hmmdefs –M hmm5 sil.hed Monesphones1

The commands in the sil script add the transitions and then create the T model. Finally two more passes of *HERest* is made on these final HMMs resulting in the final monopphone based recognizer. where *sil.hed* contains the following commands :

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

The *AT* commands adds transitions to the given transition matrices and the final *TI* command create a tied-state called *silst*. The parameters of this tied state are stored in the hmmdefs file and within each silence model, the original state parameters are replaced. One of the major problems to be faced in building any HMM based system is that of data insufficiency. To overcome this problem, HTK allows a variety of sharing mechanisms to be implemented whereby HMM parameters are tied together so that the training data is shared and more robust estimates result. This involves converting the set of initialized and trained context independent monophone HMMs to a set of context dependent models. In our experiment, this is done in two steps. Firstly, the monophone transcriptions have been converted to tri-phone transcriptions and then a set of tri-phone models is created by copying the monophones and re-estimating again.

5.6.5 Tri-phone Construction

The *triphone*1 transcription is created automatically using *HLed* tool form the *monophone* transcriptions and result is stored in a file named *wintri.mlf*. At the same time, a list of triphones is written to the file named triphones:

HLed -n triphones -i wintri.mlf mktri.led phonesI.mlf

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

WB sp

WB sil

TC

The two *WB* commands define *sp* and *sil* as word boundary symbols. The clone command *CL* in the script file below takes as its argument the name of the file containing the list of triphones and for each model of the form *a-b+c* in this list, it looks for the monophone b and makes a copy of it. The *TI* command is used to explicitly tie all members of a set of transition matrices together. *TC* command is used to convert tri-phone and outputting a list of the distinct tri-phones. The list of items within brackets is patterns designed to match the set of tri-phones, right bi-phones and left bi-phones for each phone. The following invocation performs the cloning of the models efficiently:

HHed -H hmm9/macrosI -H hmm9/hmmdefs -M hmm10 mktri.hed monophones

The file *mktri.hed* is generated using a *perl* script; the source code of which is included in once the context-dependent models have been cloned, the new tri-phone set can be re-estimated using *HERest*. This is done as previously except that the monophone model list is replaced by a tri-phone list and the tri-phone transcriptions are used in place of the monophone transcriptions as in the following:

HERest –C config1 –I wintri.mlf –S train.scp –H hmm10/macros –H hmm10/hmmdefs –M hmm11 triphones

One more re-estimation was performed so that the resultant model sets were ultimately saved in hmm12.

5.6.6 Making Tied-State Triphones

As discussed in the previous section, the first stage of model refinement is usually to convert a set of initialized and trained context independent monophone HMMs to a set of tri-phone models. The next step in the model building process is to tie states with in tri-phone sets in order to share data and thus to be able to make robust parameter estimates.
The idea of parameter *tieing* is appropriate in the case where there is insufficient training data to estimate a large number of model parameters reliably [28]. For the purpose of parameter *tieing*, *HHed* provides a mechanism that uses decision trees and is based on asking questions about the left and right contexts of each tri-phone. The decision tree attempts to find those contexts which make the largest difference to the acoustics and which should therefore distinguish clusters. Decision tree state *tieing* is performed by running *HHed* in the normal way, i.e.

HHed -H hmm12/macrosspdtiedsp1 -H hmm12/hmmdefsspdtiedsp1 -M hmm13 tree.hed triphones>log

The edit script *tree.hed,* which contains the instructions regarding which contexts to examine for possible clustering can be rather long and complex. However, it has been tried to generate this script file manually to test for the effects of *tieing.* The *tree.hed* file used in this experiment is attached as Appendix G.

Each *QS* command loads a single question and each question is defined by a set of contexts. For example, the first *QS* command defines a question called L-Class-Stop which is true if the left context is either of the stops B /b/, T /t/, D /d/, K /k/, or G /g/.
For a tri-phone system, it is necessary to include questions referring to both the right and left contexts of a phone, the questions should progress from wide, general classifications (such as consonant, vowel, nasal, diphthong, etc.) to specific instances of each phone. The second *TR* command enables intermediate level progress reporting so that each of the following *TB* commands can be monitored. Each of these *TB* commands cluster one specific set of states. For example, the first *TB* command applies to the first emitting state

of all context-dependent models for the phone a. The values given in the *RO* (100, in this experiment) which is used to set the outlier threshold and TB (375 in this case) commands affect the degree of *tieing* and therefore the number of states produced in the clustered system. The values should be varied according to the amount of training data available.

An important advantage of tree-based clustering is that it allows tri-phone models which have no training data to be synthesized. This is done *HHed* using the *AU* command which has the form:

AU hmmlist

Its effect is to take as its argument a new list of tri-phones (*hmmlist*) expanded to include all those needed for recognition and any physical models listed which are not in the currently loaded set are synthesized.

Once all state-tieing 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 tieing them together, producing a new list of models called *tiedlist*. Finally and for the last time, the models have been re-estimated twice using *HERest* changing the name of the input and output directories (set with the options -H and -M) each time *HERest* is run. The first of the two invocations is as follows:

HERest –C config –i wintri.mlf –S train.scp –H hmm13/macros –H hmm13/hmmdefs –M hmm14 tiedlist

Now that the models have been created by sufficiently training, with the necessary data set following step-by-step produces of HMMs, the next step is to evaluate the performance of the recognizer.

5.7 Speaker-Dependent Models

The same design principles have been followed in order to build the speaker-dependent models. The only and the major difference is it requires relatively small number of

speakers. In order to build this model, 4 speakers (2 female speakers and 2 male speakers) in the same age group (20-40) have been used. Moreover, 2 of them have been involved in the training process and 2 of them (1 female and 1 male) have not been involved in the training process and used to test the speaker-dependent models for previously unknown users.

5.8 Recognizer Evaluation

Recognition and Analysis are the final tasks in the process of developing a speech recognizer. In this study the performance of the recognizer is analyzed using the test data. This helps us to see how robust is the system developed in the recognition of Tigrigna speech.

5.8.1. Accuracy measurement

Results output by the ASR system have to be compared with expected correct results in order to measure system's performance. The question "Does the proposed method improve the recognition of the data?" is to be answered by measuring the performance. In this thesis, the correct word rate is measured with the formula;

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

where N denotes the total number of words in recognized sentences, D denotes deletions and S denotes substitutions. This formula does not include insertions.
Another measure is recognition accuracy, given as;

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

where I denotes insertions.
The performance of the recognizer is then determined by the tool called HResults. HResult reads in a set of label files that are output from the recognition tool HVite and compares them with the corresponding reference transcription files. The comparison is based on a Dynamic Programming based string alignment procedure.

5.8.2 Performance of continuous speaker-dependent recognition

For the phone-based recognizer HResults is invoked as follows and the table 5.2 shows the test result for phone-base continuous speaker dependent:

HResults –I testdetadep.mlf monophones1 recoutdep.mlf

	Accuracy	Ins.	Del.	Subs.
Words correctly recognized	66.30 %	0	653	78
Words accurately recognized	62.50 %	3	678	89
Sentences correctly recognized	23.05 %	0	0	14

Table 5.2 monophone tested on the testdata.

For the tied-state triphone based recognizer HResults is invoked as follows the table 5.3 shows the test result for tri-phone-base continuous speaker dependent

HResults –I testdatadep.mlf tiedlist recoutstieddep.mlf

	Accuracy	Ins.	Del.	Subs.
Words correctly recognized	64.86 %	0	51	38
Words accurately recognized	60.45 %	4	58	43
Sentences correctly recognized	21.35 %	0	0	13

Table 5.3Triphones tested on the testdata.

5.8.3 Performance of continuous speaker-independent recognition

For the phone-based recognizer: HResults is invoked as follows

HResults –k tsI%%* -I testdata.mlf monophones1 recout1.mlf

System	% Word Correct	%Word Accuracy	%Sent. Correct
M (-p=0.0, -s=5.0)	48.45	40.42	5.74
M (-p=5.0, -s=10)	56.37	54.01	16
Ins.	0	41	0
Del	722	718	0
Subs.	38	42	42

Table 5.4 The monophone system with different –p and – s values.

The recognition tool, HVite is executed once again on the monophone models changing only two of its parameters - the word insertion penalty (-p) 1 and the grammar scale factors (-s) 2.These parameters can have a significant effect on recognition performance. The –p value is incremented from 0.2 earlier to 5.0 and the –s value was taken up to 10.0 [40]. The effect of these values is displayed by the above table.

For the tied-state triphone based recognizer:

HResults –k tsI%%* -I testdata.mlf tiedlist recouttied1.mlf

	Accuracy	Ins.	Del.	Subs.
Words correctly recognized	60.32 %	0	67	48
Words accurately recognized	58.38 %	6	71	44
Sentences correctly recognized	20 %	0	0	40

Table 5.5 Triphones tested on the testdata.

where *testdata.mlf* and *traindata.mlf* are actual transcription files of the test sets against which the transcription files generated by the recognizer; namely, recouttied.mlf and recouttied.mlf are compared. The *–k s* option is used so that recognition results are displayed on a speaker by speaker basis as well as globally where s defines a pattern which is used to extract the speaker identifier from the test label file name. The character % matches any character whilst including it as part of the speaker identifier. The wildcard * is used to match zero or more characters.

The recognition errors encountered in this experiment are mainly of two kind deletion and substitution. Although insignificant, there are also insertion errors in the experiment.

The reason for deletion errors is a failure to speak louder since the recognizer ignores the input as a background noise.The second type of error is substitution errors in which an incorrect speech unit is identified by the recognizer for another input speech. Among the reasons for the substitution errors includes phonetic similarity, faulty creation of reference model (incorrect labeling and segmentation), in sufficient data, and the utterance of a particular speaker. Insertion errors are encounter when recognizers are mach extremely sensitive to sounds. The recognizers, in this care, become capable to accept very softly spoken speech inputs but also become more sensitive to environmental

noise. For such cases, the recognizer sensitivity needs to be adjusted and set at a level that requires the speakers to speak in a normal voice. The remaining insertion errors in the test data are mainly of the noise in the recording environment and from placement of a microphone. The recording environment in this experiment is not the same for all the speakers. The placement of the microphones is the other variable that is not constant. This obviously leads to shift in position of the microphone which in turn widens variability in the utterances of the speakers & even some times it generates a higher.

5.8.3 Analysis of Results

As can be observed from the results of the experiments phonemes and tied-state triphones have come up with better performance. In fact, it is natural to expect tied-state triphones to produce better recognition results as it captures most of the important contextual effects. The main reason for this relatively degraded performance may be attributed to the toolkit used in the experiment. The speaker-independent models have resulted in a very good performance for the training data using tied-state triphone models but practically speaking, insignificant improvement has been obtained for the testing data. The reasons why the results for the testing data is not as good as that of the training data is believed to be inadequate test set triphone coverage in the training set.

This can be handled by training the triphones (context-dependent models) from a large data base or corpus. Another possible reason could be lack of complete and exhaustive information about the rules of Tigrigna. All sequences of sounds are not permissible in a language [37] and it would be interesting to determine which clusters are possible in Tigrigna and which are prohibited by the rules of the language. To put it another way, it would have been useful to determine the rules of Tigrigna that decide which consonants may cluster together in which order, and which are prohibited. This is a task beyond the scope of this project, but the availability of these information would have helped this project during constructing the script file *tree.hed* which contains the instructions regarding the contexts to examine for possible clustering The study admits that the rules specified in the script file *tree.hed* are by no means exhaustive and are used simply to test the effects of tieing. It is hoped that the availability of a large database with good

phonetic balance and coverage will make both models; namely, the speaker-dependent and the speaker-independent models, more robust with the tied-state tri-phone approach.

The process of building the recognizers has been carried out in a laboratory where other student researchers are working; hence, interpersonal discussions and frequent door slams plus background noise were unavoidable. The back ground noise can affect a recognition system in two ways. In the first place, it possibly could have introduced unintended extraneous acoustic elements into the channel via the microphone. Second, speakers are not indifferent to stress and background noise [18]. Stress produces emotional and physical responses that affect the way a person speaks. Since many of the speakers are strangers to the student researcher in the room, they are not at ease and the results of some of the speakers are very low because of the observable stress they were in during recording. Moreover, some of the speakers could not speak consistently and considerable degraded performance has been observed for these speakers on the training data. Besides, since the system used flat-start training scheme, the automatic segmentation and labeling procedures normally make errors in addition to the fundamental ambiguity of the signals.

Comparison has also been made based on the number of models used in building the different recognizers. The phoneme-based recognizer is made up of 44HMM models one for each phoneme including the silence and the sp model. On the other hand, the tied-state triphone-based recognizer consisted of 44-models including the silence and sp models. These context dependent triphones are generated automatically using the HTK tool HLEd. However, some extra work had to be done in order to tie states together for the purpose of sharing data using phonetic decision trees. However the results have shown that there is no significant difference in the performance of the phoneme-based model and tied-state triphone-based model on the training data for Speaker dependent models.

CHAPTER SIX

CONCLUSIONS AND RECOMMENDATION

6.1 Conclusions

Different speech recognition techniques are being developed and implemented to generate a natural and intelligible speech as speech recognition systems are becoming more applicable. Moreover, the technologies have also advanced to incorporate speech recognition systems in different computing devices.

In this thesis work, a first attempt is done to develop speech recognition for Tigringa language using Hidden Markov Model. To come up with the new recognizer, first every component of HMM based speech recognition system is studied to identify those components that are dependent on the characteristics of a language. Having those components in mind, the Tigrigna language is studied and feature such as phonemes is used for designing prototype speech recognition.

Literature indicates that phonemes are appropriate units of recognition for continuous and speaker independent applications [15]. But it is also discussed that they are context independent units that assume identical units where ever that unit is. Actually this is not true in reality. In languages like Tigrigna, a phone may display different features as the context varies. This otherwise will have its own negative impact on recognizers performance. In this Investigation, phonemes take as base units and to buttress up the mentioned side effect, they are promoted to a left-right context sensitive units, known as tri-phones for continuous speaker independent and speaker dependant model. This has significantly boosted the performance from 54.01% word level accuracy to 58.38% word level accuracy in the triphone system; and from 16.0% to 20.0% sentence level accuracy. The word insertion penalty (-p) and the grammar scale factors (-s) had a significant impact on the performance improvement of the recognizer. However there is no significant effect on the performance for speaker dependent.

The pronunciation dictionary that we have used does not handle the problem of gemination of consonants and the irregular realization of the sixth order vowel and the

glottal stop consonant, which has a direct effect on the quality of the sub-word transcriptions. Proper editing (use of phonetic transcription) of the pronunciation dictionaries which, however, requires a considerable amount of work, certainly will result in a higher quality of sub-word transcription and consequently in the improvement of the recognizers' performance

6.2 Recommendation

For improving the performance of Tigrigna speech recognition developed in this study, we made the following recommendations.

- The corpus data should be as large as possible because in this statistical approach or in the Hidden Markov Model there are probability parameters that need to be estimated. In addition the corpus data should be recorded in various environmental conditions and from demographical linguistically distinct groups to improve performance. Hence there is a need to come up with a standardized large vocabulary speech corpus that can help to design robust speech recognizer.
- Environmental robustness can affect the performance of the recognition greatly. There is a need to design an adaptation methods that helps the system learn and readjust it self to new environments.
- In this practical work of statistical approach HMMs are used but literatures indicates there is a need to under take research using neural networks and mixed types which contain both HMM and neural networks to improve there performance.
- In this application, a conventional Hidden Markov Model is used. But a modified HMM, for Example, inclusion of separate state duration model can be explored to improve the performance of speech recognizer.
- The pronunciation dictionary that we have used does not handle the problem of gemination of consonants and the irregular realization of the sixth order vowel and the glottal stop consonant, which has a direct effect on the quality of the sub-word transcriptions. There is a need to construct a well-versatile pronunciation dictionary the various language issues evolved.

Reference

[1]. Barbara Resch (minor changes by Erhard Rank) .nd."Automatic Speech Recognition with HTK ".a Tutorial for the Course Computational Intelligence Signal Processing and Speech Communication Laboratory Inffeldgasse 16c/IIphone 873–4436. Available at: http://www.igi.tugraz.at/lehre/CI(accesed on may 2009).

[2]. Furui, Sadaoki. 2005."50 Years of Progress in Speech and Speaker Recognition Research". ECTI Transactions on Computer and Information Technology Vol.1. No.2.

[3]. Ganapathiraju, Aravind. 2002. "Support Vector Machines for Speech Recognition". A Ph.D Dissertation, Faculty of Mississippi State University, U.S.A.

[4]. Gesprochener Sprache. 1998." Rheinisch-Westfälischen Hochschule Aachen". Ph. D. Thesis. Kontinuierlich.

[5]. Girma Berhe .2001. "A Stemming Algorithm Development for Tigrigna Text Documents". MSc Thesis, School of Information Studies for Africa, Addis Ababa University, Ethiopia

[6]. Girmay, Berhane.1983"The phonology of Tigringa". Master's Thesis, Addis Ababa University, Addis Ababa,

[7]. Gao Zheng Hong and Lei Mo .2002,Speech Recognition Techniques for Digital Video Library:Ph.D thesis Department of Computer Science and Engineering The Chinese University of Hong Kong.

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

[9]. Hunt, M.J.1995. "Signal Representation" Survey of State of the Art in Human Language Technology. Eds. Cole, R. A. et al., Oregon Graduate Institute, 11-16.

[10]. Hussien Seid and Björn Gambäck. 2005." A Speaker Independent Continuous Speech Recognizer for Amharic"interspeech ,2005 pp.3349-3352

[11]. Juang, B. H. and Lawrence R. Rabiner. 2005. “Automatic Speech Recognition A Brief History of the Technology Development”. Elsevier Encyclopedia of Language and Linguistics, Second Edition.

[12]. Junqua, Jean-Claude and Jean- Paul Haton.1996.” Robustness in Automatic Speech Recognition: Fundamentals and Applications”. Boston: Kluwer Academic Publishers.

[13]. Janˇ Cernocký, Valentina Hubeika .2009.”Speech Recognition using DTW and HMM”. FIT BUT Brno23. dubna .

[14]. Kinfe Taddesse. 2001. “Sub-Word Based Amharic Word Recognition: An Experiment Using Hidden Markov Model (Hmm)” .MSc Thesis, School of Information Studies for Africa, Addis Ababa University, Ethiopia

[15]. L.R. Rabiner. 1989.” A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition”. Proceedings of the IEEE. pp. 257-285.

[16]. Lee, K-F, H-E, Hon, and R. Reddy. 1990. “An overview of SPHINX Speech Recognition System” . Readings in Speech Recognition. Eds. Waibel and K-F, Lee. 600-610.

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

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

[19]. Martha Yifiru. 2003. “Automatic Amharic Speech Recognition System to Command and Control Computers”. MSc Thesis, School of Information Studies for Africa, Addis Ababa University, Ethiopia

[20]. Molalgne Girmaw. 2004. “An Automatic Speech Recognition System for Amharic”. MSc Thesis, Dept. of Signals, Sensors and Systems, Royal Institute of Technology, Stockholm, Sweden.

[21]. Murat Ali Çömez. 2003.” large Vocabulary Continuous Speech Recognition for Turkish Using HTK”. Master’s Thesis, School Of Natural and Applied Sciences of the Middle East Technical University.

[22]. Nahm, E. and D. Slater. 1997. “Speech Recognition” .The Ultimate Multimedia Handbook. Ed. Keyes, Jessica , New York: McGraw-Hill.

[23]. Nebiyou Tsegaye , 2005 .” speech to text conversion using Amharic characters”. Master’s Thesis, Faculty of technology, Addis Ababa University.

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

[25]. Olsen, S.2003.”Fundamentals of Linguistic Analysis”. Available at http://www2.huberlin.de/angl/olsen-analysis.pdf

[26]. Philip, G and E. S. Young.1987. “Man-machine Interaction by voice: Developments in Speech Technology” .Journal of Information Science vol. 13. 3-14.

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

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

[29]. Rodman, R. D. 1999. “Computer Speech Technology”. Norwood: Artech House, Inc.

[30]. S. Young. 1996. “Large Vocabulary Continuous Speech Recognition: a Review. Technical Report”. Cambridge University Engineering Department.

[31]. S.Ortmanns. nd.” Effiziente Suchverfahren zur Erkennung”. Kontinuierlich.

[32]. Sami Lemmetty, 1999. “Review of Speech Synthesis Technology”, Master’s Thesis, Helsinki University of Technology.

[33]. Saurabh Chatterjee, Harish Karnick, Srinivasan Umesh.nd .”Speech Recognition in Indian Languages”.Btp term1 report. Available at http://www.igi.tugraz.at/lehre/CI

[34]. Solomon Berhanu. 2001.” Isolated Amharic Consonant-Vowel (CV) Syllable Recognition: An experiment using the Hidden Markov Model.” Master’s Thesis, faculty of informatics ,Addis Ababa University.

[35]. Solomon Tefera Abate, Wolfgang Menzel and Bahiru Tefla. 2005.” An Amharic speech corpus for large vocabulary continuous speech re-cognition”. In

Proc. 9th Europ. Conf. on Speech Communication and Technology, Interspeech, Lisbon, Portugal.

[36]. Solomon Teferra Abate, Martha Yifiru Tachbelie, Wolfgang Menzel.2007. "Amharic Speech Recognition: Past, Present and Future". The University of Hamburg, Department of Informatics, natural language systems division. Vogt-Kölln-str. 30, D-22527 Hamburg, Germany.

[37]. Solomon Teferra Abate. 2006. "Automatic Speech Recognition for Amharic". Ph.D. Thesis. Available at: http://www.sub.unihamburg.de/opus/volltexte/2006/2981/pdf/thesis.pdf.

[38]. Tesfaye Yihdego.2003."Diphone text to speech synthesis system for Tigrigna Language ".Master's Thesis, faculty of informatics, Addis Ababa University.

[39]. Tudor Dimitrov Ganchev. 2005 "Speaker recognition". Dissertation subject at the Patras University for the Doctor degree of philosophy, Greece.

[40]. W. H. Abdulla, N. K. Kasabov. 1999. "The Concepts of Hidden Markov Model in Speech Recognition". Technical Report. University of Otago.

[41]. Wiggers Paskal. 2001." Hidden Markov Models for Automatic Speech Recognition and their Multimodal Applications". Delft University of Technology:The Netherlands.

[42]. X. Huang, R. Reddy. 2001. "Spoken Language Processing". Pearson Education.

[43]. Young, Steve; Dan Kershaw; Julian Odell and Dave Ol-lason. 2009. The HTK Book.Young, Steve et al. The HTK Book. Microsoft Corporation.

[44]. Zaidi R , et al .2008." Quranic Verse Recitation Feature Extraction Using Mel-Frequency Cepstral Coefficient (MFCC)".Faculty of Computer Science and Informatio n Technology, University o f Malaya Department of Al-Quran & Al-Hadith, Academy of Islamic Studies, University of Malaya,

[45]. Zegaye Seifu. 2003."Hidden Markov Model Based Large Vocabulary, Speaker Independent, Continuous Amharic Speech Recognition ".Master's Thesis, faculty of informatics, Addis Ababa University.

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

[47]. Zue, Victor, Ron Cole and Wayne Wand. 1996. “Speech Recognition” In Survey of the State of the art in Human Language Technology. Edited by Ronald A. Cole, Joseph Mariani, hans Uszkoreit, Annie Zaen-en and Victor Zue. Available at: http://www.lt-world.org/HLT_Survey/master.pdf

[48]. "What is phonetics", http://www.sil.org/linguistics/GlossaryOfLinguisticTerms/WhatIsPhonetics.htm , last updated 5 January 2004, (accessed on 3/16/2009).

[49]. “ Tigrigna language” from Wikipedia, the free encyclopedia http://en.wikipedia.org/wiki/Tigrinya_language last modified on 27 May 2009 (accessed on 30 May 2009).

[50]. Stephen C. Cook.2002. http://www.faqs.org/docs/Linux-HOWTO/Speech Recognition-HOWTO.htm.

Appendix A: The Tigrigna character set

Tigrinya writing system

	Ä	u	i	a	e	(ə)	o
h	ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ
l	ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ
ḥ	ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ
m	መ	ሙ	ሚ	ማ	ሜ	ም	ሞ
s	ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ
r	ረ	ሩ	ሪ	ራ	ሬ	ር	ሮ
s	ሰ	ሱ	ሲ	ሳ	ሴ	ስ	ሶ
š	ሸ	ሹ	ሺ	ሻ	ሼ	ሽ	ሾ
k'	ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ
kʷ'	ቈ		ቊ	ቋ	ቌ	ቍ	
x'	ቐ	ቑ	ቒ	ቓ	ቔ	ቕ	ቖ
xʷ'	ቘ		ቚ	ቛ	ቜ	ቝ	
b	በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ
v	ቨ	ቩ	ቪ	ቫ	ቬ	ቭ	ቮ
t	ተ	ቱ	ቲ	ታ	ቴ	ት	ቶ
č	ቸ	ቹ	ቺ	ቻ	ቼ	ች	ቾ
h	ኀ	ኁ	ኂ	ኃ	ኄ	ኅ	ኆ
n	ነ	ኑ	ኒ	ና	ኔ	ን	ኖ
ñ	ኘ	ኙ	ኚ	ኛ	ኜ	ኝ	ኞ
ʼ	አ	ኡ	ኢ	ኣ	ኤ	እ	ኦ
k	ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ
kʷ	ኰ		ኲ	ኳ	ኴ	ኵ	
x	ኸ	ኹ	ኺ	ኻ	ኼ	ኽ	ኾ
xʷ	ዀ		ዂ	ዃ	ዄ	ዅ	
w	ወ	ዉ	ዊ	ዋ	ዌ	ው	ዎ
ʻ	ዐ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ
z	ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ
ž	ዠ	ዡ	ዢ	ዣ	ዤ	ዥ	ዦ
y	የ	ዩ	ዪ	ያ	ዬ	ይ	ዮ
d	ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ
ǧ	ጀ	ጁ	ጂ	ጃ	ጄ	ጅ	ጆ
g	ገ	ጉ	ጊ	ጋ	ጌ	ግ	ጎ
gʷ	ጐ		ጒ	ጓ	ጔ	ጕ	
t'	ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ
č'	ጨ	ጩ	ጪ	ጫ	ጬ	ጭ	ጮ
p'	ጰ	ጱ	ጲ	ጳ	ጴ	ጵ	ጶ
s'	ጸ	ጹ	ጺ	ጻ	ጼ	ጽ	ጾ
s'	ፀ	ፁ	ፂ	ፃ	ፄ	ፅ	ፆ
f	ፈ	ፉ	ፊ	ፋ	ፌ	ፍ	ፎ
p	ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ

Appendix B : Tigrigna phonemes adopted from [5]

Tigrigna symbol	Ethop font	In this paper	Tigrigna words example
ቡ	/b/	/b/	/bun/=’coffee’
ፓ	/p/	/p/	/pasta/=’pasta’
ጵ	/p´/	/P/	/p´et´ros/=’peter’
ዱ	/d/	/d/	/dimmu/=’cat’
ትሮ	/t/	/t/	/tolo/=’quick’
ጥ	/t’/	/T/	/t’af/=’tef’
ግ	/g/	/g/	/goggo/=’bread’
ክ	/k/	/k/	/kalbi/=’dog’
ቅ	/k'/	/q/	/k’al/=’word’
ኣ	/ʙ/	/a/	/ʙadgi/=’donkey’
ፍ	/f/	/f/	/fik’ri/=’love’
ዝ	/z/	/z/	/zibʙi/=’hyena’
ስ	/s/	/s/	/siga/=’meat
ሽ	/š /	/S/	/šaš /=’veil’
ፅ	/s’/	/x/	/s’om/=’fast’
ሀ	/h/	/h/	/hafti/=’wealth’
ሐ	/ħ/	/H/	/ ħaw/=’brother’
ዕ	/ɘ/	/A/	/ʕarki/=’friend’
ቸ	/č/	/C/	/ č iggir/=’problem’
ጭ	/č'/	/X/	/ č'aw/=’salt’
ጅ	/j/	/j/	/jigna/=’brave
ዥ	/ž/	/Z/	/does not exist/
ም	/m/	/m/	/ma?si/=’leather mat’
ን	/n/	/n/	/nega/=’ tomorrow’
ኝ	/ñ /	/N/	/ ñak/=’sound produced when a dog eating food’
ል	/l/	/l/	/lada/=’lamb’
ር	/r/	/r/	/ri?si/=’head’
ው	/w/	/w/	/waga/=’price’
ይ	/y/	/y/	/yeman/=’right-hand’

Appendix C: Pronunciation dictionary (phone-based dictionary)

A A sp
AESlnetu A E S l n e t u sp
AESnetdo A E S n e t d o sp

AEbdan A E b d a n sp

AEbuynet A E b u y n e t sp

C C sp
E E sp
.
.
wAElom w A E l o m sp

wHudatn w H u d a t n sp

wSTen w S T e n sp

wala w a l a sp

xaAErii x a A E r ii sp

xbaH x b a H sp

xbq x b q sp

xbuq x b u q sp

xbuqe x b u q e sp

ybten y b t e n sp

ydelii y d e l ii sp

ydgem y d g e m sp

zwqeT z w q e T sp
zxebeqet z x e b e q e t sp
zxenHe z x e n H e sp
zxenHo z x e n H o sp
zzkra z z k r a sp

Appendix D: Grammar files (The task Grammar)

(sil(zteAaxaxefe|xaAErii|nlmAat|mesno|lomii|Aamet|tejemiiru|zelon|kulu|maHbereseb|ge
Ter|bzleAale|menen|nahrn|zsatefelu|zelon|haftii|tefeTro|nay|mHwayn|mknKann|medeb|qe
ndii|AElamu|Eyu|haftii|tefeTro|nay|maHwayn|mknKann|sraH|ms|kulu|zefer|lmAat|geTer|z
tewadeden|zteasseren|zeytnkfo|Aawdii|lmAat|yelen|AEwt|sraHtii|AEqebe|Hamedun|mayn
|bmokyad|gobotat|bagrabet|zSefenelu|kunetat|EntfTer|meTen|maykers|mdrn|lAElii|meTen
|ywseK|Aiisran|arbaAEten|Tqmtii|klte|Siihn|arbaAEten|Entmerxenii|seleste|negerat|nmfxe
m|qal|atye|neyre|Etii|Hade|Halefney|bzgbaE|kfxm|Entezeykiile|kab|Halafnetey|kemzleq|q
uxre|Enssa|zegedemn|aAEwafn|ybezH|bu|abiilu|kebabyawii|nbret|ayer|ymaHayeS|nmrba
H|Enssa|zebietn|anahbn|zmCu|kunetat|yfeTer|Telii|zAEqebu|kunetat|sle|zmeCaCo|meTen
n|teketatalnetn|znab|EnawesKe|ymexE|baHafeSu|tefeTro|ktberkto|zgbaa|SeSey|bzeyquTe
ba|Ethbelu|AEdl|bzyade|ymeCaCe|haftii|tefeTro|ms|lmAat|mesno|zteassru|mkanu|mgnzab|
ygbaE|qdmii|kulu|kXbeT|zelewa|qum|neger|netii|nlmAat|mesno|wesnay|zKone|hafiitii|ma
y|nay|mAEqabn|mgulbatn|guday|Eyu|ab|lAElii|meriet|zelo|maywn|lkAE|kemtii|nmesleTy
a|sreH|tebahiilu|zmdeb|bejet|bagbabun|bquTaben|mTqam|wHlnet|yKewn|ksab|Hzii|temek
uro|kemzegenzbena|bmesno|klemAE|zgboa|sfiiH|kebabii|Enahalew|btegbar|gna|keylemAe
|zxenHe|mKanu|Eyu|tbxaH|nab|EsenadaEtii|gazieTa|weyn|Eza|zsdelKum|zeleKu|nuSetey|
gTmii|nnbab|abxHulay|ab|qexalii|tesatafay|gazieTaKum|kem|zKewn|Ewn|qal|yatu|ztewad
ede|xaAErii|nab|zleAale|Aawet|zebxH|Eyu|nKulom|Aaqmtat|ab|mAEwat|lmAat|mesno|kn
ewAElom|ygbaE|ab|Hder|klte|Siihn|Hade|Aamete|mHret|ab|zwexet|mexhiet|bzeAEba|amb
asder|kas|gebre|hywet|bzuH|tegenziibna|ane|Etii|qedem|zenbebkwam|mexHaftii|rusiiya|le
kas|bambasader|kas|Eyum|tetergomom|ksab|Hzii|nezii|keyfeleTku|maxnHay|Haziine|zkon
e|koynu|lomii|mexhiet|weyn|nezii|slezefeleTetnii|amesgn|nambasader|kas|dma|egenAE|be
luley|slezkone|qaley|slezeyfexemku|zhabkumnii|Halafnet|meliise|nKerekbekum|wesiinii|a
laKu|prozdent|mesgagere|mengstii|somaliiya|nebere|Aabdiilahii|yosuf|kab|slTanom|mwre
dom|nporlama|abzeflaTlu|Ewan|zbelwa|lomii|Etii|bzuH|geltaAEtaAE|teriifu|netii|hzbawii|
teqalasy|nay|zedHane|sdre|tariiK|knknbto|iina|dHrii|wHudat|meAaltat|Etom|qolAu|teleqii
qomo|wezero|lkie|EKele|may|EnaqWaxera|marie|ztebahlet|nESetey|geza|dma|Enatemelale
sa|maHbHaben|qexela|ms|iihwedieg|bekal|Enteytelaleyu|bfqrii|iihwedeg|teleKefa|iihwedie
g|xlal|wxAat|mkwanu|bzeytentane|terediiEwam|bEmnet|xnAatn|Etii|sdra|medergtii|yeblun
|wezero|lkie|zbelana|knekafalekum|waga|Emnet|Entay|mkwanu|ngenzeb|ane|netii|tegadela
y|bAayney|ayreeKwan|kulu|gzie|gn|melEKatii|ylEkelay|neyru|mKad|EyKEln|wala|Enateq
enzewe|mxHafn|mnbabn|ayeqWarexn|neyru|Ezii|zemen|Ezii|msHalefa|knrekeb|iina|kHalf
a|dma|Eyu|Enabele|melEKtii|ylEKeley|neyru|yeAiintii|dergawyan|kab|mereba|ayte|zeriihu
n|knqela|aykealen|ayte|zeriihun|qetrii|qetrii|meAalom|feqedo|boreKa|neyru|maHres|besiiru
|deleyie|nay|densii|welfii|slezHazo|eykonen|nmejemerya|gze|sle|zegaTom|neyru|ndens|Elu
|kHata|ydelii|Enbii|EntebeletniiKe|wrdet|Eyu|maryu|teseE|dense|Enabele|yXqXqo|neyru|b
rhane|gn|mEntii|densii|kdns|eydeleyien|brhane|bwSTu|teXeneqe|Hade|wedii|mexu|nktdns|
Hateta|nab|brhane|qulH|iila|des|keybela|tesiia|kedet|brhane|kemzwenena|nmntay|qedemeni
i|iilu|Hareqe|Eta|Etejemeret|muziiqa|ksab|EtwedeE|tehaweKe|brhane|bTaAEmii|tebasaXe
w|wxu|keKyd|ayxelen|Etii|porto|kwdaE|Enateqarebe|Eyu|brhane|gn|ksab|Hezii|aydenesen|
Aaynii|Aaynu|tTmt|neyra|nay|meXereSa|dansii|slezKone|deqii|Enstyo|ms|zmerexeo|wedii|
kdnsa|elewn)sil)

Appendix E: File HSparse(SLF)

# Define size of network N=number of nodes and L=number of arcs
N=1814 L=3619 #List of nodes: I= node -number , W=word

I=0 W=!NULL
I=1 W=!NULL
I=2 W=sil
I=3 W=zteAaxaxefe
I=4 W=!NULL
I=5 W=xaAErii
I=6 W=nlmAat
I=7 W=mesno
I=8 W=lomii
I=9 W=Aamet
I=10 W=tejemiiru
I=11 W=zelon
I=12 W=kulu
I=13W=maHbereseb
I=14 W=geTer
I=15 W=bzleAale
I=16 W=menen
I=17 W=nahrn
I=18 W=zsatefelu
I=19 W=zelon
I=20 W=haftii
I=21 W=tefeTro
I=22 W=nay
I=23 W=mHwayn
I=24 W=mknKann
I=25 W=medeb
I=26 W=qendii
I=27 W=AElamu
I=28 W=Eyu
I=29 W=haftii
I=30 W=tefeTro
I=31 W=nay
I=32W=maHwayn
I=33 =mknKann
I=34 W=sraH
I=35 W=ms
I=36 W=kulu
I=37 W=zefer
I=38 W=lmAat
I=39 W=geTer
I=40 W=ztewadeden
I=41 W=zteasseren
I=42 W=zeytnkfo
I=43 W=Aawdii
I=44 W=lmAat
I=45 W=yelen
I=46 W=AEwt
I=47 W=sraHtii
I=48W=AEqebe
I=49W=Hamedun
I=50 W=mayn

List arcs : J=arc-number, S=Start-node , E=end-node

J=3550 S=2
E=1745
J=3551 S=2
E=1746
J=3552 S=2
E=1747
J=3553 S=2
E=1748
J=3554 S=2
E=1749
J=3555 S=2
E=1750
J=3556 S=2
E=1751
J=3557 S=2
E=1752
J=3558 S=2
E=1753
J=3559 S=2
E=1754
J=3560 S=2
E=1755
J=3561 S=2
E=1756
J=3562 S=2
E=1757
J=3563 S=2
E=1758
J=3564 S=2
E=1759
J=3565 S=2
E=1760
J=3566 S=2
E=1761
J=3567 S=2
E=1762
J=3568 S=2
E=1763
J=3569 S=2
E=1764
J=3570 S=2
E=1765
J=3571 S=2
E=1766
J=3572 S=2
E=1767
J=3573 S=2
E=1768
J=3574 S=2
E=1769
J=3575 S=2
E=1770
J=3576 S=2
E=1771
J=3577 S=2
E=1772
J=3578 S=2
E=1773
J=3579 S=2
E=1774
J=3580 S=2
E=1775
J=3581 S=2
E=1776
J=3582 S=2
E=1777
J=3583 S=2
E=1778
J=3584 S=2
E=1779
J=3585 S=2
E=1780
J=3586 S=2
E=1781
J=3587 S=2
E=1782
J=3588 S=2
E=1783
J=3589
S=2E=1784
J=3590
S=2E=1785
J=3591
S=2E=1786
J=3592 S=2
E=1787
J=3593 S=2
E=1788
J=3594 S=2
E=1789
J=3595 S=2
E=1790
J=3596 S=2
E=1791
J=3597 S=2
E=1792
J=3598 S=2
E=1793
J=3599 S=2
E=1794
J=3600 S=2
E=1795
J=3601 S=2
E=1796
J=3602 S=2
E=1797
J=3603 S=2
E=1798
J=3604 S=2
E=1799
J=3605 S=2
E=1800
J=3606 S=2
E=1801
J=3607 S=2
E=1802
J=3608 S=2
E=1803
J=3609 S=2
E=1804
J=3610 S=2
E=1805
J=3611 S=2
E=1806
J=3612 S=2
E=1807
J=3613 S=2
E=1808
J=3614 S=2
E=1809
J=3615 S=2
E=1810
J=3616 S=2
E=1811
J=3617 S=2
E=1812
J=3618 S=4
E=1813

Appendix F: The Prototype Definitions for the Phoneme-based Approach

```
~o <VecSize> 39 <MFCC_0_D_A>
~h "protoI"
<BeginHMM>
  <NumStates> 5
  <State> 2
    <Mean> 39
      0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
    <Variance> 39
      1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
  <State> 3
    <Mean> 39
      0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
    <Variance> 39
      1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
  <State> 3
    <Mean> 39
      0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
    <Variance> 39
      1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
  <State> 4
    <Mean> 39
      0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
    <Variance> 39
      1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0

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

Appendix G: The File tree.hed

```
RO      100.0 stats

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+*,*^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_Fortis"               {*^s+*,*^k+*,*^t+*,}
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_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_Fortis"       { *^t-*,*^k-*,*^s-*,*^f-*,*^P-*,*^K-*}
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_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_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_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_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_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_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_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_b_4_" {("b","*-b+*","b+*","*-b").state[4]}
TB 350 "ST_a_4_" {("a","*-a+*","a+*","*-a").state[4]}
```

N

```
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_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 "fulllist"
CO "tiedlist"
ST "trees"
```

O

Declaration

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

Declared by:

Name: _____

Signature: _____

Date: _____

Confirmed by advisor:

Name: _____

Signature: _____

Date: _____

Place and date of submission: Addis Ababa, November 2009