Synthetic Speech Trained - Large Vocabulary Amharic Speech Recognition System (SST-LVASR)

A thesis Presented to the School of Graduate Studies

Addis Ababa University

In Partial Fulfillment of the Requirements for the Degree of Master of Science in Communication Engineering

By

Mesfin Birile Woldetsadik

Addis Ababa, Ethiopia

July 2008

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

FACULTY OF TECHNOLOGY

Synthetic Speech Trained - Large Vocabulary Amharic Speech Recognition System (SST-LVASR)

By

Mesfin Birile

APPROVAL BY BOARD OF EXAMINERS

Dr.Mengesha Mamo	_____
Chairman Dept. of Graduate Committee	Signature
Dr. V.N.V Manoj	_____
Advisor	Signature
Mr. Molalgne Girmaw	_____
Advisor	Signature
Dr. Eneyew Adugna	_____
Internal Examiner	Signature
Dr. Worku Alemu	_____
External Examiner	Signature

Abstract

Amharic is the official language of Ethiopia, which is characterized by very large morphological forms of words. This thesis is an investigation of the possibility of developing an Automatic speech recognition system (ASR) for Amharic using synthesized Amharic speech generated through concatenation of prerecorded morphemes, can be used to train a hidden markov model (HMM) based ASR system.

The development of HMM based ASR system requires identification of all possible words and a construction of text and speech corpora containing multiple samples of the words to be recognized by the system. These data are then used as training sets in the development of the models, the final objective being the construction of HMM models for each recognition unit. Since there are a large number of morphological forms for the words in Amharic, the effort of collecting the Amharic words for constructing the text corpus and the recording and labeling of the same words for the speech corpus is extremely difficult. This thesis demonstrates that by developing an automatic morphological expander, the effort of developing the text corpus is reduced to a manageable level. Additionally, a significant reduction in the speech corpus development is achieved by using machine generated speech for training the HMM models of the ASR system. These reductions in the development efforts of the text and speech corpora greatly reduce the most prominent of the obstacles in developing a general purpose Amharic speech recognizer.

The 62.37% word accuracy for naturally recorded speech indicates that using synthetic speech for training at least 62% of the words are correctly identified and suggests that with synthetic speech some level of recognition is possible, giving the imputes for more research in finding ways to increase this accuracy.

Acknowledgment

I would like to extend warmest thanks to my advisor Ato Molalgne Girmaw who has given me endless support, motivation and discussion for the successful completion of this thesis work.

I am also grateful to my family, friends and all Physics and Engineering Department staffs for their moral, constructive comments and material support.

Acronyms

ANN: Artificial neural network

ASR: Automatic speech recognition

DCT: Discrete cosine transforms

EBNF: Extended Bakus Naur Form (grammar specification language)

FFT: Fast Fourier transforms

MSSTATE: Mississippi State

CSLU: Center for Spoken Language Understanding

HTK: The Hidden Markov Toolkit

LPC: Linear prediction coding

LVASR: Large vocabulary automatic speech recognition

MFCC: Mel frequency cepstral coefficients

PLP: **P**erceptual linear prediction

TTS: Text to Speech synthesis

SST-LVASR: Synthetic Speech Trained –Large Vocabulary Amharic Speech Recognition

HSLAB: HTK module for manuplating speech lable files

XFST: Xerox Finite State Tools

Terminologies

Cepstrum: A representation of speech signals that is found by taking the Fourier transform of the log-magnitude spectrum of the speech waveform.

Grapheme: A symbol used in a written language

Morpheme: The minimal meaning-bearing unit in language

Phonemes: Basic sounds used in a language.

Syllabary: A set of written symbols that represent (or approximate) syllables, which make up words.

Contents

List of Figures

List of Tables

Chapter 1 Introduction

Speech is a natural way of communication among people. Speech communication refers to the processes associated with the production and perception of sounds used in spoken language. We learn to speak prior to write or type. Although there are many professional typists, it is not practical to employ them for all of our typing needs. Automatic speech recognition (ASR) is a useful alternative form of input. It has several advantages over using a computer keyboard. ASR can be used while someone is doing some work or looking somewhere. It also allows people with handicaps to use computers. ASR reduces injuries that arise from using computer keyboards for a long time or from concentrating on the computer screen. ASR can also aid in education; an example is its use as a language-learning tool.

For many languages speech recognition systems have been developed with some success. A notable example is Microsoft Office XP's speech recognition system. Other more specific ASR systems have also been used in different applications with continuing success.

A general-purpose speech recognition system (a dictation system) can be used to encode documents into text. This may be especially useful for languages that have a very large character set. Large characters set languages are difficult to type using keyboards, as they require a number of keystrokes to type for a single character. For such languages typing is usually a very time consuming process. Amharic is an example of this type of languages.

1.1. Introduction to ASR systems

Automatic speech recognition (sometimes referred to as just speech recognition, computer speech recognition or erroneously as voice recognition) is the process of converting speech signals uttered by speakers into a sequence of words, which they are intended to represent, by means of an algorithm implemented as a computer program. The recognized words can be the final results, as for applications such as data entry and dictation systems or the words so recognized can be used to trigger specific tasks as in command and control applications. Speech recognition systems can be catacorized based on different parameters [5], some of the more important of which are shown in Table 1.1.

Parameters	Range
Speaking Mode	Isolated words vs. continuous speech
Speaking Style	Read Speech vs. Spontaneous Speech
Enrollment	Speaker-dependent vs. Speaker-independent
Vocabulary Size	Small (less than 20 words identified) vs. Large (greater than 20,000 words identified)
Language Model	Finite-state vs. Context-sensitive
Signal to Noise Ratio (SNR)	High (> 30dB) vs. Low (< 10dB)

Table 1.1: Typical parameters used to characterize the capability of speech recognition systems

An isolated-word speech recognition system requires that the speaker pause briefly between words, whereas a continuous speech recognition system does not. Spontaneous speech is much more difficult to recognize than speech read from script. Some systems require speaker enrollment – a user must provide samples of his or her speech before using them, whereas other systems are said to be speaker-independent, in that no enrollment is necessary. Some of the other parameters depend on the specific task.

1.2. Applications of ASR systems

Speech recognition applications vary depending on the vocabulary size, the quality of the microphone, the number of users, and the tolerance for error of the users [4]. A few examples include:

- Hands-free control of machinery:
  - Small vocabulary,
  - Speaker-independent,
  - High-quality microphone (often a headset microphone),
  - Very low error tolerance (error tolerance can be increased with verbal feedback).
- Automatic telephone dialing
  - Small vocabulary,
  - Mixture of speaker-independent (digits) and speaker-dependent (names),

- Low-quality microphone (telephone handset),
- Moderate error tolerance (if the system asks for confirmation before dialing).

- Telephone access to databases
  - Moderate vocabulary,
  - Speaker-independent,
  - Low-quality microphone,
  - High error tolerance.
- Word processing
  - Large vocabulary,
  - Speaker-dependent,
  - High-quality microphone,
  - High error tolerance.

1.3. The Amharic Alphabet

The Amharic alphabet consists of thirty-four basic characters, each of which has six additional modified characters. The modified characters represent the basic sound of the symbol augmented with a vowel. Thus, the main table of the traditional Amharic syllabary, appears as characters set in thirty-four rows and seven columns. Languages using such a scheme have been termed to use an “Abugida” instead of an alphabet [12]. Abugida is a term coined by Peter Daniels for a script whose basic signs denote consonants augmented with a vowel and where consistent modifications of the basic sign indicate augmentation of other vowels. The term “Abugida” is derived from the first four characters of one type of ordering of the Ge`ez script. The Amharic script is an abugida, although the vowel modifications in Amharic are not entirely systematic.

As a result of using an abugida, each concatenated consonant-vowel syllable in the Amharic language has its own corresponding orthographic symbol. A few syllables, however, are exceptions to this rule. These syllables are represented by more than one symbol in the character set. These redundant orthographic symbols used to make differences in sound realization as well

as meaning in old Amharic. In modern Amharic, these differences do not exist.

Another implication of the use of abugida is that there are a large number of characters in the language. This makes encoding Amharic using a computer keyboard difficult. Using current Amharic text encoding programs and typewriters, it routinely takes 2 or 3 keystrokes to type in a single character. It is not unheard of to find characters that take 4 keystrokes on some of the Amharic text encoding programs. It is instructive to compare this with English that requires only a single keystroke for all its characters except for certain special characters and capital letters in which case only two keystrokes are necessary.

1.4. The Challenge of Speech Recognition for Amharic

Speech recognition is generally a difficult task. Firstly, the mapping from symbols to speech is not one-to-one since different underlying symbols can give rise to similar speech sounds. Furthermore, there are large variations in the realized speech waveform due to speaker variability, mood, environment, etc., and the boundaries between symbols cannot be identified explicitly from the speech waveform. No two realizations of the same speech will be identical even when uttered by the same speaker one after the other. Every consecutive utterance of the same word will be different. Noise and channel distortion also present a great challenge in the recognition process.

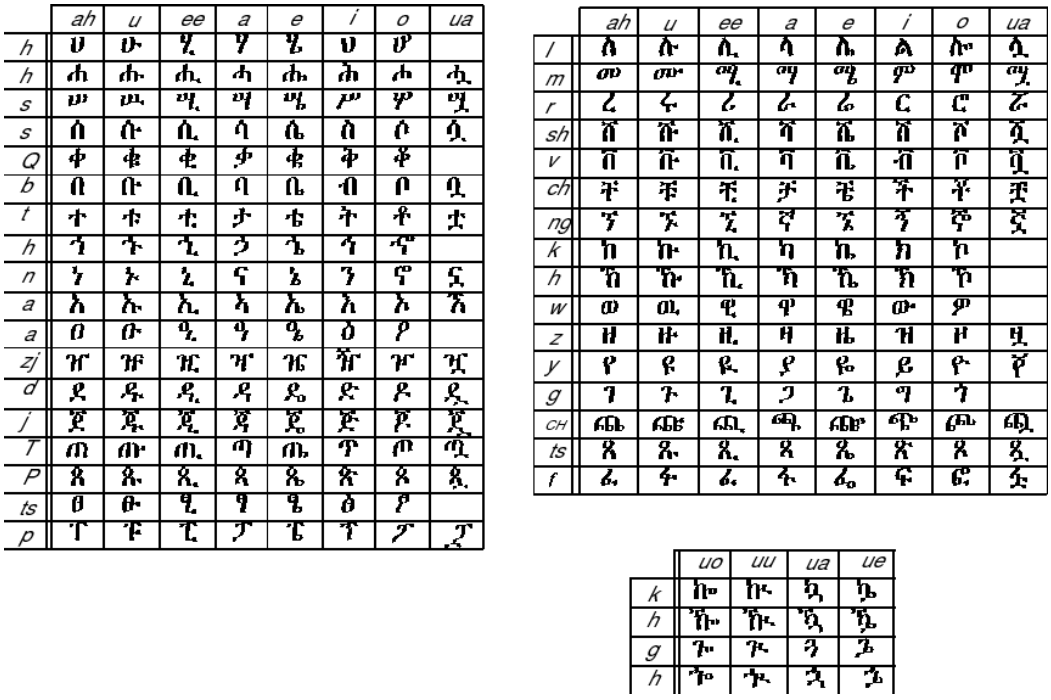

Figure 1.1: The Amharic Abugida

The first problem of different symbols arising from the similar speech waveforms is not as pronounced in Amharic as in other languages such as English. This is a direct result of the systematic representation of speech sounds by a consistent set of symbols in the Amharic abugida. As a result mapping speech waveforms to orthographic symbols is not as difficult as (at least in theory) other languages. The Amharic orthography consists of 276 distinct symbols excluding the twenty numeral symbols and the eight punctuation marks [19]. The main task of speech recognition is concerned with identifying distinct speech units, thus as a decent approximation the writing of Amharic the redundant symbols may be taken out of the set of symbols in the speech recognizer without losing the essential understanding in the text. This results in a total of only 233 graphemes (syllabic characters) in the language.

Amharic, on the other hand, presents other challenges not seen in languages like English. One such, and very important, difficulty is the number of distinct words in the Amharic vocabulary. In Amharic, each basic word has many more morphological forms as compared to, for example, English. This is exemplified by the number of morphological forms of verbs. In Amharic, there are much more than one thousand morphological expansions for each verb. As far as speech

recognition is concerned, each of these words are considered distinct and must be considered separately, raising the complexity of Amharic speech recognition thousands of times more than that of English. In other words, assuming that the number of base words in English and Amharic are roughly equal, for every word that English ASR systems must worry about, there are at least 1000 words for which Amharic ASR systems must worry. Assuming that there are 20,000 base words in both languages and assuming there are on average five morphological forms for English words, while the English ASR system must account for 100,000 distinct words, its Amharic equivalent needs to account for 20,000,000 words.

The first challenge is collecting these words and constructing a text corpus and the next challenge is to record a sufficiently large number of speech samples of each of these words as they appear naturally in spoken sentences. These recorded samples must then be labeled to indicate the start and end of the phones. Since a single sample of a word is not sufficient to build the models multiple samples of the word are required. Assuming that 10 samples of each word are required, the speech corpus development for English would require 1,000,000 labeled samples of the words, while Amharic would require 200,000,000. Furthermore, if the system is to be speaker independent, the experiment need be repeated with multiple speakers. If we again assume 10 speakers are used. English would in total require 10,000,000 labeled speech utterances, while Amharic would require 2,000,000,000. Although some of the assumptions might not be correct, the level of complexity for English and Amharic is clearly obvious.

A further challenge with regards to Amharic is that not all of these words are known. There is, up to this time, no definite resource which has collected all possible words of Amharic. This, however, is not true for English; as such corpora have already been developed. Thus, even if one wants to record the speech samples, one must start from collection of the words. This makes, the development of the text and speech corpora for Amharic a difficult, time consuming and expensive effort and is one of the reasons why Amharic speech recognition system development is a very challenging task.

1.5. Statement of the Problem

Speech recognition is a difficult problem, largely because of the many sources of variability associated with speech signals. First, the acoustic realizations of phonemes, the smallest sound units of which words are composed, are highly dependent on the context in which they appear. Second, acoustic variability can result from changes in the environment as well as in the position and characteristics of the transducer. Third, within-speaker variability can result from changes in

the speaker's physical and emotional state, speaking rate, or voice quality. Finally, differences in socio linguistic background, dialect, and vocal tract size and shape can contribute to across-speaker variability [5].

Recognition is generally more difficult when vocabularies are large or have many similar-sounding words. When speech is produced in a sequence of words, language models or artificial grammars are used to restrict the combination of words.

Perhaps the biggest challenge in developing an Amharic ASR is the development of the text and speech corpora. The Amharic vocabulary is very large mainly as a result of the large number of morphological forms each words have. In Amharic morphology, each word can be expanded into a large number of forms. If we consider the case of a verb we notice that it has more than one thousand different forms (compare this with English, which has just four derived forms for most of its verbs (e.g. smoke, smoking, smokes, smoked). Verbs, in Amharic can take many forms to indicate time (past, present, future, etc), gender (male, female), action (command, statement, invitation etc). The verbs also change forms depending on the gender of the subject and object. In effect, a single modified verb can be used to express complex sentences in other languages like English. For example the single modified Amharic verb (ተጠራሩ (Teteraru)) means “They called each other”. As a result of the Amharic morphological structure each word has a large number of derived forms, resulting in a very large vocabulary.

Current ASR technology demands that all these word to be stored in a dictionary and an acoustic model be developed for each of these words. The development of these models in turn requires that multiple speech realizations of each of these words be available. To make the final ASR system speaker independent, each of these speech samples must be recorded using different speakers from a select group of demographically representative population. What this means, in effect, is that, for each of these words, at least several recorded samples be available. These samples must then be labeled indicating the start and end of each of the phones that make up the word.

The implication is quite simple, if there are 100,000 words, first of all these words must be identified and stored in a dictionary (develop the text corpus). Secondly some one million-speech sample must be collected and labeled (develop speech corpus). The effort to accomplish this task is clearly very big and this is the main reason why researchers have not been able to develop a full Amharic speech recognizer.

1.6. Goals, Objectives and Expected Outcome of the Thesis

1.6.1. Objectives

The objective of this thesis is to investigate the possible use of machine synthesized speech based on morphological rules in the development of a large vocabulary continuous Amharic speech recognition system.

1.6.2. Goals

The goals of this thesis are

- Developing and implementing an algorithm to simplify the text corpus development based on morphological expansion rules of Amharic,
- Developing and implementing an algorithm to simplify the speech corpus development by automatically synthesizing and labeling training sentences by first concatenating prerecorded morphemes of words (prefix, suffix, and stem word) to generate the possible morphological forms of words and concatenating these words in appropriate sequence to generate the training sentences,
- Developing a hidden markov model based large vocabulary automatic speech recognizer for Amharic using these synthesized sentences as training speech data,
- Testing the developed ASR system on human speech to see if acceptable accuracy is achieved.

1.6.3. Expected Outcome

The expected outcome of the thesis is an algorithm that minimizes the effort of development of an Amharic speech synthesizer by minimizing the development effort of both the text and speech corpora. If, at the end of the thesis, good recognition accuracy is achieved the algorithm has effectively reduced the ASR development effort by a factor of thousands, which value depends on the exact number of morphological forms of the words.

1.7. Solutions Approach

The development of ASR systems can be conveniently divided into two major stages – language modeling and acoustic modeling. Each of these stages requires a data preparation stage, which are referred to as the text corpus and speech corpus development stages, respectively. We can, therefore, separate the development of an ASR system into four stages. Normally, the text corpus is developed by collecting each individual word in the vocabulary of the language and the speech corpus is developed by recording and labeling sentences containing multiple occurrences of each of the words in the text corpus.

In this thesis, the development of both the text and speech corpora is automated. For the case of the text corpus, only the base words in the vocabulary of the language are collected and the derived words are generated automatically using a morphological expansion program, which will be designed and implemented in the course of the thesis. For the case of the speech corpus, recording and labeling is performed for all base words and a few selected sample morphological forms of certain words. The collection of sentences in the corpus is then made in a two-stage process. First, the morphologically derived words of the majority of the words are generated by concatenating the recorded base words with the required morphemes that is extracted from one of the recorded words containing these morphemes. Special care is taken to minimize the effect of co-articulation by considering left and right contexts of the morphemes to match the target words' contexts and the original contexts.

The development of the language model for SST-LVASR could be either stochastic or deterministic. In this thesis, deterministic language modeling is chosen, simply because it is easier to develop a sample language model. The scope of this thesis does not include the development of a proper language model. Therefore no attempt is made to make the language model complete or even partially representative of Amharic. The sole reason the language model is used is to make the ASR system work, as ASR systems essentially require some language model to perform their recognition tasks. The language model in this thesis will simply involve one type of acceptable sentence in Amharic. This, of course, limits the type of sentences that can be recognized by the system. Nevertheless, it will be sufficient for the purpose of the investigation.

The acoustic modeling proceeds in exactly the same way as standard ASR systems approach it. The speech corpus is used to extract a set of feature vectors, which are used to train HMM models for each recognition unit - initially by taking a global average for all monophones then

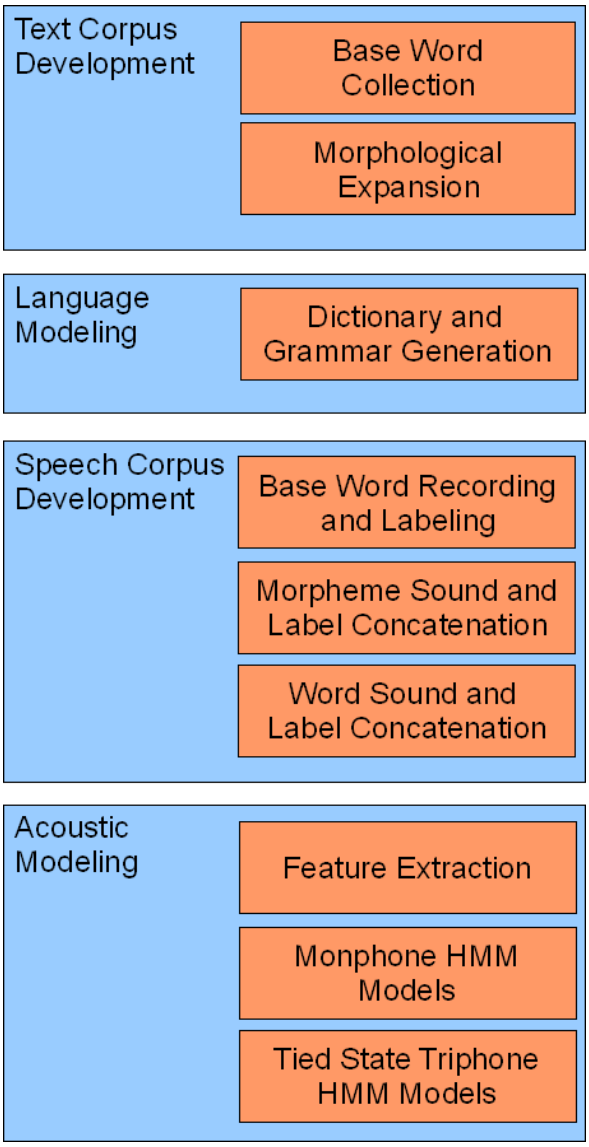

Figure 1.2: The four stages of SST-LVASR development.

successively refining these averages by applying the Baum-Welch algorithm. These models are then further refined by extending the modeling to triphones and then by creating tied state between the triphone models to represent acoustic similarities. These four stages of SST-LVASR development are summarized in Figure 1.2.

1.8. Outline

Chapter 2: Literature Review

Chapter 2 describes about the theory of ASR systems, hidden markov models (HMM), speech synthesis, morphological analysis, the use of synthetic speech in speech recognition, and prior works on Amharic ASR and Morphology.

Chapter 3: Text Corpus Development

Chapter 3 describes in detail the approach taken in this research thesis to develop the text corpus required by speech recognizers. This approach is based on morphological expansion of Amharic words and uses scripting and automatic generation of the morphological derivation of words recognizer units and hidden markov models. It also incorporates the methodology used in the development of the text and speech corpora, the morphological expander algorithm, the development of training data, data labeling, training, testing and evaluation for the Amharic LVASR with Synthetic Training Speech.

Chapter 4: Speech Corpus Development

Chapter 4 describes in detail the approach taken in developing the training speech corpus for the ASR system. In this thesis, the speech corpus is based on concatenation of prerecorded morphemes that are extracted from a recording of words. These morphemes are concatenated to form a word speech corpus, which itself is used for generating of the full sentence corpus.

Chapter 5: Implementing the SST-LVASR System

Chapter 5 describes the implementation of the ASR system using the HTK toolkit. The steps taken to achieve the final recognizer using the text and speech corpora developed in chapter 3 and 4 are described.

Chapter 6: Recognizer Evaluation and Results Analysis

Chapter 6 is dedicated to the description and analysis of the results obtained form the *SST-LVASR.*

Chapter 7: Concluding Remarks

Chapter 7 summarizes the thesis work, describes the limitations, and recommends improvements and future work and possible applications of the outcomes of the research work.

Chapter 2 Literature Review

The aim of this study is the investigation in the use of synthetic speech as a training data in the development of large vocabulary Amharic speech recognizer system. The speech is to be generated using morphological rules of Amharic word expansion. Thus the research focuses on three main aspects of natural language processing: word morphology, speech synthesis and automatic speech synthesis. The literature review, therefore, focuses on these three aspects. Furthermore, prior work on Amharic speech recognition and morphology is discussed as the research is interested in the application of these concepts for Amharic.

2.1. Theory of ASR Systems

The basic objective of a speech recognition system is to take a speech signal and convert it into a sequence of words. Unfortunately, speech technology has not matured well enough to unambiguously extract the sequence of words from the speech signal. Instead, all the current speech recognizers use a probabilistic model of speech recognition, which inevitably introduces some errors. The science of speech recognition, at its current state focuses in reducing these errors to an acceptable level for the required application. The objective of speech recognition in these probabilistic models can be stated as "**Given a speech signal *S* find the most likely word sequence *W* represented by the signal**".

Speech signal is an air pressure wave emanating from the mouth and nostrils to communicate thoughts. These continuous variations in air pressure are perceived by the human auditory system and interpreted into their intended meaning. Using this natural speech-understanding scheme as a starting point, a source-channel mathematical model can be formulated for speech recognition.

The meaning associated with the speech sound sequences is dependent upon the spoken language in use. Each language specifies a certain sequence of speech sounds to represent words. The basic building blocks of spoken words representing a distinct speech sound are called phonemes. The phonemes combine to form syllables and syllables combine to form words and words are combined in sequence to construct meaningful statements.

2.2. The Fundamental Statistical Model of Speech Recognition

The verbal expression of the speech recognition problem, which was stated previously and repeated below, can be expressed in mathematical form using equation (2.1). Given a certain speech signal *S* select the word sequence *W* for which the probability is largest

$$P(W/S) = \frac{P(W)\,P(S/W)}{P(S)} \qquad (2.1)$$

Equation (2.1) expresses the probability of the word sequence *W* given the speech signal *S* using Bayes' theorem. After recording a particular speech signal *S*, it is desired to select the word sequence *W* for which the probability of observing *W* given that the speech waveform *S* has been observed, i.e. *P(W/S)* is maximal. Bayes's theorem expresses this probability in terms of the probability of the observed speech signal *S* (which is fixed for all word sequences compared during the determination of the maximum probability), the probability of a particular word sequence (which depends only on the language and not on the uttered speech), and the probability of observing the speech signal *S* given the word sequence is known to be *W*.

The reason why Baye's theorem is used here is because of the difficulty of computing directly the probability *P(W/S)*. So far no known algorithm can give a closed form expression for computing this probability. Therefore, all statistical algorithms compute this probability indirectly using Baye's theorem. Thus, a successful statistical algorithm needs to compute the two probabilities *P(W)* and *P(S/W)* for all possible candidate words. The third probability, viz., *P(S)* does not play a role in the speech recognition problem. The reason is simple, since *P(S)* is fixed for all candidate words (the speech is already uttered and known), it does not play a part in the determination of the word *W* with the maximum value of *P(W/S)*. Therefore, no models are required to estimate its probability *P(S)*. Thus, in practice, finding the sequence of words that maximize the following expression can solve the recognition problem.

$$P(W/S) = P(W)\,P(S/W) \qquad (2.2)$$

The probability *P(W)* is a function of the language. It specifies how frequently the candidate word *W* occurs as compared to all other words in the language's vocabulary. For that reason, it is referred to as the *word* probability. The algorithm that calculates this probability is called the *language model*.

The probability *P(S/W)* describes the probability that the uttered speech *S* will result from realizing the candidate word *W*. In other words, it describes how probable it is to realize the speech waveform *S* while uttering the candidate word *W*. Since the realization of speech does not have a one-to-one correspondence with that of a word that it represents, this probability tries to identify the most probable word among the possible candidate words that has resulted in the realization of the speech. It is not very difficult to intuitively imagine that the words *red* and *read* will have similar waveforms while the words red and yellow will have differing waveforms. Therefore, the calculation of these probabilities will result in a close score for red and read and large differences between *red* and *yellow*. This probability, *P(S/W)*, is referred to as the *acoustic probability* and its corresponding algorithm is called the *acoustic model*.

The objective of ASR systems therefore can be stated as the calculation of the the word and acoustic probabilities, *P(W)* and *P(S/W)*, automatically. To obtain a solution to this problem in an actual speech recognition system, several issues must be addressed. First of all the speech signals need to be represented somehow in a format amenable for computer processing, secondly we need to break down the probabilities into further smaller probabilities to reduce the data requirement and computational complexity, finally algorithms must be developed to compute both the language probability and the acoustic probability.

2.3. Speech Recognizer Units

In many speech recognition systems phonemes are used as the basic unit of recognition. Phonemes represent the basic sounds that are made in the language. An actual realization of a phoneme is called a phone. Each phoneme has a particular pronunciation in the language. This does not mean that the actual speech waveform generated for a phoneme is invariably identical. The waveform generated depends on many factors including the speaker, environment, and the context in which the utterance is made. In fact, it is impossible that any two realizations of a phoneme will be identical, but the characteristic of the generated waveforms will have strong correlation and this can be used in speech recognition.

Phonemes however are not the only possibilities to be used as recognition units, a number of

other linguistically or acoustically based units including words, syllables and even multiple words can also be used [2]. The choice of a particular recognition unit depends on the application at hand.

Speech recognizers based on words have an advantage over phone-based systems in terms of recognition accuracy. Whole-word models can capture the speech articulation differences in phonemes caused by the influence of neighboring phonemes known as co-articulation. The problem with whole-word based systems is that for large vocabulary speech recognition a large amount of training data is required. This makes the task of development virtually impossible. For small vocabulary applications, however, whole-word based models perform much better than phoneme based models. An alternative, which can serve as a compromise between the complexities of developing a word based recognizer and the co-articulation inaccuracy of phoneme-based recognizers is a syllable-based recognizer.

Syllables are less affected by co-articulation than phonemes and there are not as many as words, thus they provide a good compromise. For languages like English even syllables are not an option, since they have large number of syllables. English, for example, has more than 20,000 different syllables. Amharic, on the other hand, has only 233 distinct syllables, if we consider the only possible syllables are those that appear in Amharic Abugida. Therefore, a recognition system based on syllables is promising for Amharic [12].

Speech recognition eventually must end up in identifying words and not sub-word units like phonemes or syllables. Thus an ASR system based on phonemes must be able to convert identified phonemes into words. This can be achieved by concatenating the identified phonemes and constructing the words. Generally, each identification of phonemes is probabilistic and different candidate phonemes will arise with different probabilities. These phonemes are concatenated to form candidate words, some of which will become unacceptable as there is no such word in the vocabulary of the language. The probability of the valid concatenations (words) can be computed as the product of the probabilities of the component phonemes. In this manner, the phoneme probabilities are converted to word probabilities.

2.4. Triphone Modeling

A triphone model is a phonetic model that takes into consideration both the left and the right neighboring phones. If two phones have the same identity but different left and right context, they are considered different triphones. We refer to the different realizations of a phoneme as

allophones. Triphones are an example of allophones.

The left and right context used in triphones, while important, are only two of many important contributing factors that affect the realization of a phone. Triphone models are powerful because they capture the most important co-articulatory effects. They are generally much more consistent than context independent phone models. However, as context dependent models generally have increased parameters, trainability becomes a challenging issue when using triphones.

2.5. Speech Signal Representation

Speech is a sound wave of continuously varying air pressure. Representation of this air pressure variation in a form accessible to computers is fundamental in the analysis of speech. For that purpose the speech signal is first converted into an electrical signal using a microphone and then transformed into a discrete signal-using analog to digital converters.

For speech recognition applications the signal undergoes further processing stages. Most processing algorithms perform spectral analysis over a window containing a number of the speech samples. The window is then moved in increments of a few milliseconds. These windows are called *analysis frames*. Short time Fourier transforms are applied on analysis frames to obtain what are known as the *spectral coefficients* of the speech signal for the frame. The spectral coefficients are an alternative form of representing the speech signal in the analysis frame. The resulting coefficients form a *feature vector* for the frame. The feature vectors are used as representatives of the speech signal within that frame and are used in the computation of the acoustic probability mentioned earlier. To capture changes in the signal, a feature vector can additionally contain features of neighboring frames [14].

Several different ways of extracting the features of speech can be used. Almost all of them are based on the source-filter model of speech production. This speech production model assumes that speech is generated through of a process of filtering a source signal. Figure 2.1 shows the basic source filter-model for speech signal [1].

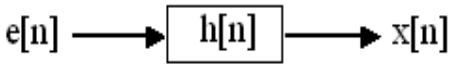

Figure 2.1: Basic source- filter model of speech production

The source signal is provided by a stream of air pushed out of the lungs through the vocal tract. This stream of air vibrates the vocal cords for voiced phonemes or gets to the oral or nasal tract unhindered for unvoiced phonemes. This is modeled by choosing the source signal **e[n]** to be either random noise or a periodic pulse to mimic unvoiced and voiced phonemes, respectively.

The stream of air that comes out of the lungs through the vocal tract is then modified by the oral and/or nasal tract. To articulate a particular phoneme, these tracts change their shape of geometry. This is achieved by changing the relative position of the various articulatory organs in the vocal and nasal tract, which include the tongue, the lips, the velum and the oral cavity. The effect of this modification is to change the resonance frequencies, which are referred to as *formant frequencies*, of the nasal and oral tracts. The formant frequencies of each phoneme are different and hence the final articulation of each phoneme will have its own formant frequency profile. In the source filter model of speech production, the last process is modeled through a filter. The source signal is fed to a filter and the output of this filter **x[n]** will be a realization of the desired phoneme (or longer speech segments, if longer durations are taken and the filter parameters are time dependent). Obviously, each phoneme will have its own filter parameters **h[n]** to reflect the particular formant frequencies corresponding to the shape of the vocal and nasal tracts.

The most obvious way of modeling each phoneme is, therefore, to choose the parameters of the filter **h[n]** as the feature vectors. There are several candidate filter models, the most common ones of which include, linear prediction coefficients, line spectral frequencies, cepstral coefficients and mel frequency cepstral coefficients.

2.5.1. Linear Predictive Coding (LPC)

Linear Predictive Coding (LPC) is one of the most powerful speech analysis techniques and one of the most useful methods for encoding good quality speech at a low bit rate. It provides accurate estimates of speech parameters, and is relatively efficient for computation.

LPC starts with the assumption that the speech signal is produced by a buzzer (voiced sounds) at the end of a tube. The glottis (the space between the vocal cords) produces the buzz, which is characterized by its intensity (loudness) and frequency (pitch). The vocal tract (the throat and mouth) forms the tube, which is characterized by its resonance frequencies, which already referred to as formant frequencies or simply formants.

LPC analyzes the speech signal by estimating the formants, removing their effects from the

speech signal, and estimating the intensity and frequency of the remaining buzz. The process of removing the formants is called inverse filtering, and the remaining signal is called the residue.

The numbers that describe the formants and the residue can be stored or transmitted somewhere else. The basic principle here is that the formants and the residues are sufficient to represent the speech signal and this information can be used to re-synthesize the speech waveform. LPC synthesizes the speech signal by reversing the process: use the residue to create a source signal, use the formants to create a filter (which represents the tube), and run the source through the filter, resulting in the original speech waveform. Because speech signals vary with time, this process is done on short chunks of the speech signal – the analysis frames. Usually 30 to 50 frames per second give intelligible speech with good compression. LPC is one of the methods to compute both the source, excitation or residue, **e[n]** and the filter (formant), **h[n]** from the speech signal **x[n]**.

The basic task of the LPC system is the determination of the formants and the source from the speech signal. The solution is obtained by solving a difference equation, which expresses each sample of the signal as a linear combination of previous samples. Such an equation is called a *linear predictor*, thus the name Linear Predictive Coding. The LPC models the formants with an all pole filter as shown in equation (2.3).

$$H(Z) = \frac{X(Z)}{E(Z)} = \frac{1}{1 - \sum_{K=1}^{P} a_K Z^{-K}} = \frac{1}{A(Z)} \tag{2.3}$$

Where *P* is the order of the LPC analysis. The inverse filter *A(z)* is defined as:

$$A(Z) = 1 - \sum_{K=1}^{P} a_K Z^{-K} \tag{2.4}$$

Taking inverse z-transform of equation (2.3) results in

$$X[n] = \sum_{K=1}^{P} (a_K X[n-k]) + e[n] \tag{2.5}$$

In LPC, the present sample is predicted as a linear combination its past *p* samples which yields

$$\tilde{X}[n] = \sum_{k=1}^{P} (a_k X[n-k]) \quad (2.6)$$

The prediction error when using the approximation of equation (2.6) is given by

$$e[n] = X[n] - \tilde{X}[n] = X[n] - \sum_{k=1}^{P} (a_k X[n-k]) \quad (2.7)$$

The coefficients a_k of the difference equation (the prediction coefficients) characterize the formants, so the LPC system needs to estimate these coefficients. The estimate is done by minimizing the mean-square error between the predicted signal and the actual signal. This is a straightforward problem, in principle. In practice, it involves the computation of a matrix of coefficient values, and the solution of a set of linear equations. Several methods (autocorrelation, covariance, recursive lattice formulation) may be used to assure convergence to a unique solution with efficient computation.

2.5.2. Mel Frequency Cepstral Coefficients (MFCC)

Mel Frequency Cepstral Coefficients (MFCCs) are, much like LPC, coefficients that can be used to represent speech signals. They are derived from a type of cepstral representation of the speech signal (a "spectrum-of-a-spectrum"). The difference between the cepstrum and the mel-frequency cepstrum is that in the MFC, the frequency bands are positioned logarithmically (on the mel scale) which approximates the human auditory system's response more closely than the linearly-spaced frequency bands obtained directly from the Fast Fourier Transform (FFT) or Discrete Cosine Transform (DCT). This can allow for better processing of data, for example, in audio compression.

The Mels scale in terms of other frequencies is defined as:

$$m = 1127.01048 \ln\left(1 + {}^{f}/_{700}\right) \quad (2.8)$$

$$f = 700\left(e^{m}/1127.01048 - 1\right) \qquad (2.9)$$

Where m : is in mel, perceptual scale of pithes (melody) and

f : is in Hz.

MFCCs are commonly derived using the following procedure

- Take the Fourier transform of a small analysis frame of the signal
- Map the log amplitudes of the spectrum obtained above onto the mel scale, using triangular overlapping windows.
- Take the Discrete Cosine Transform of the list of mel log-amplitudes, as if it were a signal.
- The MFCCs are the amplitudes of the resulting spectrum.

MFCCs are often used in speech recognition systems. They are also increasingly finding uses in music information retrieval applications such as genre classification, audio similarity measures, etc.

MFCC values are not very robust in the presence of additive noise, and so some researchers propose modifications to the basic MFCC algorithm to account for this; e.g. by raising the log-mel-amplitudes to a suitable power (around 2 or 3) before taking the DCT, which reduces the influence of low-energy components

2.5.3. Perceptual Linear Predictive Coding (PLPC)

The perceptual linear predictive (PLP) technique uses three concepts from the psychophysics of hearing to derive an estimate of the auditory spectrum.

- the critical-band spectral resolution,
- the equal-loudness curve, and
- the intensity-loudness power law.

The auditory spectrum is then approximated by an autoregressive all-pole model.

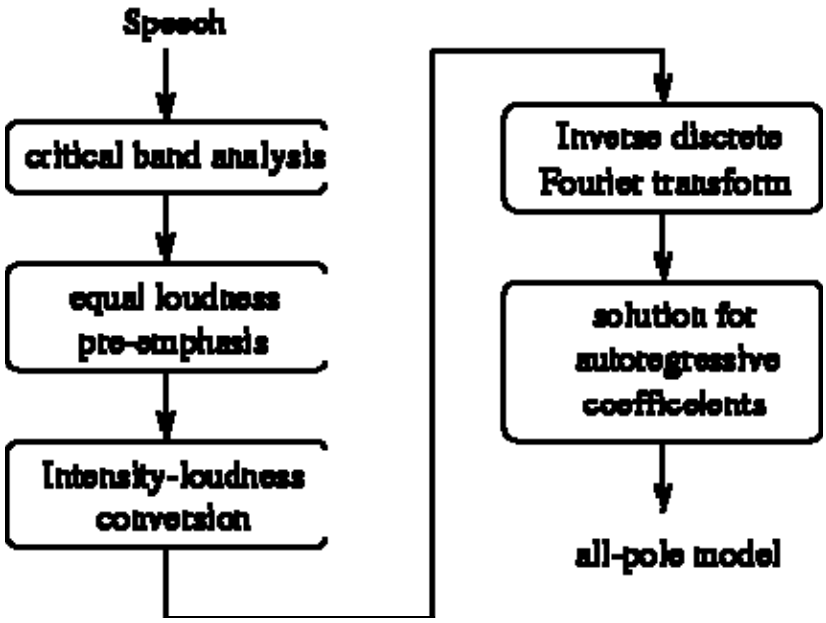

Figure 2.2: Perceptual Linear Prediction Coding

A 5th-order all-pole model is effective in suppressing speaker-dependent details of the auditory spectrum. In comparison with conventional linear predictive (LP) analysis, PLP analysis is more consistent with human hearing. PLP analysis is computationally efficient and yields a low-dimensional representation of speech. These properties are found to be useful in speaker-independent automatic-speech recognition. A linear predictive all-pole model is used to develop corresponding formants and bandwidths to which the cepstral coefficients are mapped by using a separate multiple regression model for each of the five-formant frequencies and five formant bandwidths. The dual analysis produces both the cepstral coefficients of the PLP model for the different vowel-like sounds and their true formant frequencies and bandwidths. The separate multiple regression models developed by mapping the cepstral coefficients into the formant frequencies and formant bandwidths can then be applied to cepstral coefficients determined for subsequent speech to produce corresponding formants and bandwidths used to synthesize that speech. Since less data are required for synthesizing each speech segment than in conventional techniques, a reduction in the required storage space and/or transmission rate for the data required in the speech synthesis is achieved. In addition, the cepstral coefficients for each speech segment can be used with the regressive model for a different speaker, to produce synthesized speech corresponding to the different speaker.

2.6. Language Modeling

Knowledge about language is important in recognizing and understanding natural speech. Lexical knowledge (i.e., vocabulary definition and word pronunciation) is required, as are the syntax and semantics of the language (the rules that determine what sequences of words are grammatically well-formed and meaningful). In addition, knowledge of pragmatics of language (the structure of extended discourse, and what people are likely to say in particular contexts) can be important to achieving the goal of spoken language recognition and understanding systems. In practice, in speech recognition, it may be impossible to separate the use of theses different levels of knowledge, since they are often tightly integrated [1].

In particular, language models are important to determine the probability of a particular word *P(W)* which makes up an important half of equation (2.2). Language models are important not only in speech recognition but also in many other natural language processing applications such as machine translation, part-of-speech tagging, parsing and information retrieval.

The word sequence probability is a function of the language in question. Thus, a study of the language must be made to have an accurate description of the probabilities of word sequences. Knowledge of the vocabulary, pronunciation, syntax, semantics and usage help in estimating these probabilities. Two approaches are commonly used in language modeling (*deterministic* and *stochastic* modeling).

2.6.1. Deterministic Language Models

In deterministic language modeling a formal grammatical specification is used to represent the language. A grammar is a system of rules that define the structure of a language. The formal specification of the grammar then allows the speech recognizer to analyze a recognized sequence of words and check if it complies with the grammar. For small vocabulary applications with simple rules, deterministic language models are warranted.

A deterministic language model specifies the grammar of a language in a two-step process. In the first step the words making up the language are grouped into classes of words. In the second step, the possible combination and ordering of these classes of words are defined using some type of grammar definition language. The most common method of grammar definition language is the *Extended Bakus Naur Form (EBNF).*

This process of using the EBNF can be illustrated using a simple telephone calling system. In a

telephone calling system a user can request the system to dial/call a specific number or can request it to call/dial by telling it the name of the person to be called. In this application, there are three classes of words: names, digits and commands. In the first step, the model defines these classes of words and associates all words with one of these classes. This can be done as follows.

```
$names = Abebe [Belachew] | Frehiwot [Assefa] | Netsanet [Girma];
$digit = zero | one | two | three | four | five | six | seven | eight | nine;
$command = dial | call | phone;
```

Figure 2.3: Defining the word classes in EBNF

The vertical bars "**|**" indicate an *or* operation, while the square brackets "**[**" and "]" indicate that the word enclosed between them is optional. Once this is defined, the next step is to define the possible word arrangements in the grammar. This is done as follows.

```
(start ($command <$digit> | $command $name) end)
```

Figure 2.4: Defining possible word sequences in EBNF

The two parts together form the deterministic language model. The "**`start`**" and "**end"** keywords indicate the start and end of a sentence respectively. The two angle bracket "<" and ">"indicate that the elements of the digit word class can be repeated in any sequence and length.

The use of deterministic language models in large vocabulary systems is difficult to the point of near impossibility for two reasons. First, large vocabulary systems use a complex set of grammatical rules, which are not always possible to collect and specify as a set of rules. Second, deterministic language models do not provide mechanisms that can model pragmatics - the structure of extended discourse, and what people are likely to say in particular contexts.

2.6.2. Stochastic Language Models

The most popular language model in use today is a stochastic model called the n-gram model. Stochastic language models describe the language using probabilities. They are especially useful for large vocabulary systems for which complete specification of the grammar is difficult or

impossible. An n-gram model expresses the probability of observing a word sequence of *m* words $W = w_1, w_2, \ldots, w_m$ using the previous *m-1* observations. It is built on the assumption that the probability of a specific word occurring in some text can be estimated from the frequency of its joint occurrence with other sequence of words in some given training text. The probability is given as a product of *m* conditional probabilities:

$$P(W_1, W_2, \ldots, W_m) = \prod_{i=1}^{m} P(W_i / W_1, W_2, \ldots, W_{i-1}) \qquad (2.10)$$

Equation (2.8) says that the probability of the word sequence $W=W_1,W_2,\ldots,W_m$ composed of *m* words is the product of the probability of the last word to occur given that the previous *m-1* words has already occurred and the probability of the previous *m-1* words occurring in that sequence. In other words, it assumes that the occurrence of the last word occurring depends on all previous *m-1* words, irrespective of the value of *m*. This model, however, can be simplified if the language is assumed ergodic, i.e., the probability of the last word depends only on some finite number of words $n \geq 1$ irrespective of the rest of the words that have previously occurred. With this assumption equation (2.8) is simplified to

$$P(W_1, W_2, \ldots, W_m) = \prod_{i=1}^{m} P(W_{i-n+1} / W_1, W_2, \ldots, W_{i-1}) \qquad (2.11)$$

Due to data sparsity, however, the usable ranges of *n* vary between 1 and 4 inclusive. The most common value for *n* is 3, in which case the language model is called a tri-gram model.

This model must be built using training text data called a training corpus. The algorithm looks for n joint occurrences of the words in the training corpus and calculates their probabilities, which can be used to compute *P(W)* in equation (2.2). To model a spoken language, with sufficiently accurate statistical significance, it typically requires having training corpus with many millions of words [8].

2.7. Acoustic Modeling

A number of well-known factors determine the accuracy of automatic speech recognition; those most noticeable are variations in context, in speaker, and in environment. Acoustic modeling plays a critical role in improving accuracy and is arguably the central part of any speech recognition system [1].

In a statistical framework, an inventory of elementary probabilistic models of basic linguistic units (e.g., phonemes) is used to build word representations. A sequence of acoustic parameters, extracted from a spoken utterance, is seen as a realization of a concatenation of elementary processes described by models like the Hidden Markov Models (HMMs).

The computation of the acoustic probability is commonly performed using Artificial Neural Networks (ANN) or Hidden Markov Models (HMM) or a combination of both.

2.7.1. The Hidden Markov Model (HMM)

The Hidden Markov Model (HMM) is a well known stochastic model most widely used for estimating the acoustic probability *P(S/W)*. HMM uses states and transitions to describe a class of random variables. The output sequence of the process is governed by the probability of each state to produce an output symbol and by the probability of making a transition from one state to another. Each state, therefore, has a probability distribution to produce the possible output symbols and another probability distribution to describe the likelihood of taking a transition to another state. A complete specification of a Hidden Markov Model thus requires

- An output observation alphabet $S=(s_1,s_2,\ldots,s_m)$
- A set of states $\Omega = (1,2, \ldots, N)$
- An output probability distribution associated with each state that can be described by the output probability matrix $B=f(b_{i(k)})$, where $b_{i(k)}$ is the probability of outputting symbol *k* when state *i* is entered.
- A state transition probability distribution for each state, which can be described using the so called transition probability matrix $A =f(a_{ij})$, where a_{ij} is the probability of taking a transition from state *i* to state *j*.
- An initial state probability $\pi = f(\pi_i)$, which describes the probability of starting with

state *i*.

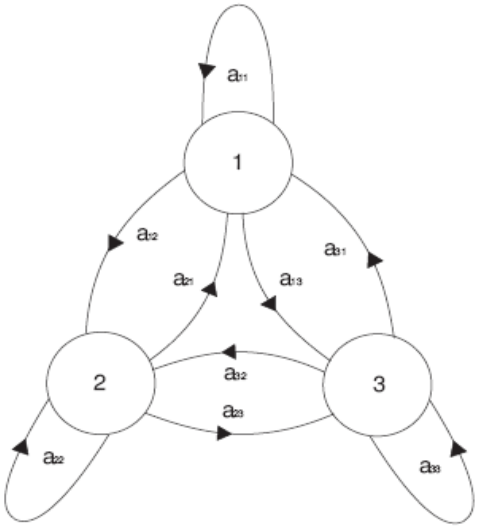

Figure 2.5: An HMM Model with three states

From equation 2.2 we see that the speech recognition problem is that of computing the word sequence having the maximum probability of occurring given the speech signal and prior probabilities of word sequences for the language. The a-priori probability *P(W)* is dependent only on the language. It is fixed even before any speech is uttered and is estimated from the behavior of the language. The remaining task is that of maximizing *P(S/W)*. It is impractical to directly compute P(S/W) = $P(s_1,s_2,\ldots,s_m /W)$ since the size of the speech parameter vectors will be too large. If the HMM model for word production is used this big task is replaced by the much simpler task of estimating the HMM model parameters.

In the HMM model for speech recognition, the speech feature vectors are assumed to be generated by an HMM process. This means that the output observation alphabet of the HMM is comprised of all the available feature-vectors. When a sequence of speech vectors $S= s_1, s_2, \ldots, s_n$ is generated the underlying HMM process undergoes a state transition. In general, any particular feature-vector sequence of length *n* can be generated by any sequence of states of the same length. Therefore there are many different state sequences associated with any particular feature vector sequence. The particular state sequence that generated the output

sequence is unknown for an external observer.

The probability of the speech vector sequence $S = s_1, s_2, \ldots, s_n$ being generated by a particular state sequence X = 1, 2,…, n is given by (assuming we know the initial state, thus neglecting the initial state probability π).

$$P(S, X) = a_{12} b_1(s_1) a_{23} b_2(s_2) \ldots a_{n-1} b_{n-1}(s_{n-1}) \quad (2.12)$$

Where a_{ij} represents the probability of the state transition from state *i* to state *j*, and $b_i(s_j)$ represents the probability of vector *j* being generated in state I. The total probability of the vector sequence can then be calculated by summing up its probabilities over all the state sequences.

Before the recognizer can perform the task of matching, a set of HMM models must be set up for each word in the vocabulary of the language. Each word has a particular set of HMM model parameters, i.e., the output and transition probabilities. The task of the speech recognizer is then to find the model that is most likely to generate the speech vector sequence *S*. During the speech recognition process, the probability of generating the vector sequence by a particular model is computed. This is done for all models. At this point we have managed to compute the probabilities *P(S/W)* for all words in the vocabulary of the language. Each of these probabilities are then multiplied with their word probability *P(W)* as computed by the language model to obtain *P(W/S)* as specified in equation (2.2). The word corresponding to the model that gives the highest probability *P(W/S)* is taken to be the solution to the speech recognition problem.

The total probability of *P(W/S)* can be used as a measure of comparison between models. However computing this total probability for each of the models is a computationally expensive task, since several state sequences of the HMM model have some finite output probability for the word and the total probability is the sum of each of these output probabilities. Therefore, in all practical recognizers, only the probability of the most likely state sequence is computed and used as a basis of comparison. An efficient algorithm called the Viterbi algorithm computes the probability of the most likely state sequence in a fraction of the computation time it takes to compute the total probability.

Three basic problems must be addressed to use the HMM model described above [2], [8]. These are:

- **The Evaluation Problem:** Given a model with known parameters and a sequence of observations, what is the probability that the model generates the observations?
- **The Learning Problem:** Given a model topology and a set of observations, how should the model parameters be chosen so that the probability of the observations being generated by the model is maximized?
- **The Decoding Problem:** Given a model with known parameters and a sequence of observations, what is the most likely state sequence in the model that produces the observations?

During the speech recognition process these three issues inevitably arise and efficient algorithms must be used to solve them. Fortunately, there are three such algorithms designed to solve the problems.

2.7.1.1. The Forward Algorithm

The goal of the forward algorithm is to compute the probability *P(S/W)*, i.e., to determine how likely an observed sequence of feature vectors is to be generated by the model representing the word sequence *W*. Therefore, the probability of observing the sequence up to a given feature vector is computed over all possible state sequences leading to that state.

The forward algorithm [2] presents an efficient way of solving the evaluation problem. However, measures need to be taken to avoid floating point underflow. This can happen if the product over several small probabilities leads to a value below the resolution of the data type in the program.

Algorithm: The Forward Algorithm

Step 1: Initialization

$$\alpha_1(i) = \pi_i b_i(S_1) \quad 1 \leq i \leq N \tag{2.13}$$

Step 2: Induction

$$\alpha_t(j) = \left[\sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij}\right] b_j(S_j) \qquad 2 \leq t \leq T \quad 1 \leq j \leq N \tag{2.14}$$

Step 3: Termination

$$P(S) = \sum_{i=1}^{N} \alpha T(i) \tag{2.15}$$

For speech recognition we may use the forward algorithm and compute the total probability of a word sequence given the speech vector sequence. Since this procedure is computationally intensive, in practice an approximation is used. This approximation computes the most likely state sequence for a given sequence of feature-vector observations. The computation of the most likely state sequence can be done in a fraction of the time required to compute the total probability. It is possible to find the most probable sequence of states for the observed features sequence by listing all possible sequences of states that result in an output of the observed sequence of vectors. We can then select the state sequence having the highest probability. This approach requires a lot of computing time.

2.7.1.2. The Baum-Welch Algorithm

The HMM modeling technique described in the previous section assumes that the output and state transition probabilities are known for each model. Generally, the HMM parameters are generated from training data with the objective of representing the words in the best possible way. This is the problem that was previously defined as the learning problem. The learning problem is solved using a popular algorithm that adjusts the HMM parameters by training them using feature vectors generated from utterances of known word sequences. This algorithm is called the *Baum-Welch Algorithm.*

The Baum-Welch algorithm is based on maximum likelihood optimization criteria through which the model probability for the words in the vocabulary of the language is generated. An

initial guess of the parameters is used for initialization before applying the algorithm. A good initialization procedure facilitates the convergence of the model parameters. Several techniques can be used to initialize the models. For example, one can choose representative examples of the phones and seed the models with the corresponding feature data. This procedure, however, is a time-consuming process requiring linguistic/phonetic expertise [16]. One simple, but effective technique is to compute the global mean and variance from the training data and set the model parameters to these values. This technique is referred to as *flat-start modeling* and is employed by the popular recognition systems. Once the models are seeded with initial values, more accurate parameters can be found by applying the Baum-Welch re-estimation formula. The algorithm is then applied successively until convergence. To apply the Baum-Welch algorithm, we define several probabilities.

The *backward probability* is defined as:

$$\beta_t(i) = P(S_{t+1}^T / S_t = i) \qquad (2.16)$$

$\beta_t(i)$ is the probability of generating the partial observation S_{t+1}^T / S_t (from $t + 1$ to the end) given that the HMM is in state i at time t.

Algorithm: calculating $\beta_t(i)$ inductively

Initialization:

$$\beta_T(i) = \frac{1}{N} \qquad 1 \leq i \leq N \qquad (2.17)$$

Induction:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(S_{t+1}) \beta_{t+1}(j) \quad t = T-1, T-2, \ldots, 1 \qquad 1 \leq i \leq N \tag{2.18}$$

Next the probability of taking the transition from state *i* to state *j* at time *t* is defined as $\gamma_i(i,j)$:

$$\gamma_t(i,j) = P(S_{t-1} = i, S_t = j / S_1^T) = \frac{\alpha_{t-1}(i) a_{ij} b_j(S_t) \beta_t(j)}{\sum_{k=1}^{N} \alpha_T(k)} \tag{2.19}$$

Using Lagrange multipliers method it can be shown that maximization of the likelihood of the speech signal *S* is achieved by computing the model parameters using this formula.

$$\hat{a}_{ij}(i,j) = = \frac{\sum_{t=1}^{T} \gamma_t(i,j)}{\sum_{t=1}^{N} \sum_{k=1}^{N} \gamma_t(i,k)} \tag{2.20}$$

$$\hat{b}_j(k) = \frac{\sum_{t s_t = o_k} \sum_{t=1}^{T} \gamma_t(i,j)}{\sum_{t=1}^{T} \sum_{i} \gamma_t(i,j)} \tag{2.21}$$

Where O_k represents the k^{th} observation alphabet and α is the forward probability computed in the same manner as the forward algorithm.

Algorithm: Baum-Welch Training Procedure

i. Begin with some set of model parameters (perhaps random, perhaps preselected),

ii. Run through the current model to estimate the expectations of each model parameter,

iii. Change the model parameters to maximize the values of the paths that are used a lot (while still respecting the stochastic constraints) using equations 2.20 and 2.21,

iv. Repeat until convergence to the optimal values for the model parameters.

2.7.1.3. The Viterbi Algorithm

The Viterbi algorithm [2] is a recursive search algorithm that finds the most likely state sequence by picking up and remembering the best path instead of summing up probabilities from all possible different state sequence paths. It is a very efficient method of computing the most likely sequence for speech recognition.

Algorithm: The Viterbi Algorithm

Step 1: Initialization

$$V_1(i) = \pi_i b_i(S_1) \qquad 1 \leq i \leq N \tag{2.22}$$

$$B_1(i) = 0 \qquad 1 \leq i \leq N \tag{2.23}$$

Step 2: Induction

$$V_t(j) = max_{1 \leq i \leq N}[V_{t-1}(i)a_{ij}]b_j(S_t) \qquad 2 \leq t \leq T \quad 1 \leq j \leq N \tag{2.24}$$

$$B_t(j) = argmax_{1 \leq i \leq N} [V_{t-1}(i)a_{ij}] \qquad 2 \leq t \leq T \quad 1 \leq i \leq N \tag{2.25}$$

Step 3: Termination

$$P = max_{1 \leq i \leq N} [V_T(i)] \tag{2.26}$$

$$ST_T = argmax[B_T(i)] \tag{2.27}$$

Step 4: Backtracking

$$ST_t = B_{t+1}(ST_{t+1}) \qquad t = T-1, T-2, \ldots, 1 \tag{2.28}$$

$$ST = (ST_1, ST_2, \ldots, ST_T) \tag{2.29}$$

ST gives the best state sequence.

2.7.2. ANN (Artificial Neural Network)

Artificial neural networks (ANN) or hybrid hidden markov Models (hybrid HMM) are interconnected group of artificial neurons that are universally known as one of the most powerful nonlinear methods for pattern recognition, time series prediction, optimization and forecasting. [7].

A neural network is a set of small computation units connected by weighted links. The network is given a vector of input values and computes a vector of output values. The computation proceeds by each computational unit computing some non-linear function of its input units and passing the resulting values on to its output units. [6]

Hybrid HMM/ANN, introduced in the nineties for speech recognition, is presently a very competitive alternative to HMM, both in terms of performances and recognition accuracy. HMM/ANN can provide discriminative training, are capable of incorporating multiple input sources, and have a flexible architecture that can easily accommodate contextual inputs and feedbacks. Furthermore, ANNs are typically highly parallel and a regular structure, which makes them especially, suited for high-performance architectures and optimized implementations.

Although artificial neural networks have been shown to be quite powerful in static pattern classification, their formalism is not very well suited to addressing automatic speech recognition (ASR). In fact in ASR there is a time dimension that is highly variable and difficult to handle directly in ANN.

The ASR problem recomposed in terms of ANNs can be stated as follows: how can an input sequence (e.g., a sequence of spectra or spectral derived coefficients) be properly classified into an output sequence (e.g., sequence of phonemes, words or sentences) when the two sequences are not synchronous, since there usually are multiple inputs associated with each phoneme or word. Several neural network architectures have been developed for (time) sequence classification [7], including:

- Static networks with an input buffer to transform a temporal pattern into a spatial pattern,
- Recurrent networks that accept input vectors sequentially and use a recurrent internal state that is a function of the current input and the previous internal state,

- Time-delay neural networks, approximating recurrent networks with feed-forward networks.

In the case of ASR, all of these models have been shown to yield good performance (sometimes better than HMM) on short isolated speech units. By their recurrent aspect and their implicit or explicit temporal memory they can perform some kind of integration over time. However, neural networks by themselves have not been shown to be effective for large-scale recognition of continuous speech. To overcome this problem some authors introduced, in the early nineties, a new approach that combine ANN and HMM for large vocabulary continuous speech recognition.

2.7.3. Speech Corpus Development

The ultimate objective of a speech recognition system is designing a system that recognizes speech when it is spoken by anyone in a natural way and in any environment. However, this omnipotent system has eluded researchers so far. Different systems using different techniques have been implemented with varying degree of successes. To achieve better accuracy and meet resource constraints placed on the system less versatile designs are employed.

- speaker independent vs speaker dependent,
- large vocabulary vs small vocabulary
- isolated vs continuous speech

Irrespective of the type of system, training speech data is required to create the acoustic models. The difference lies with the amount and variety of data that must be collected for the different systems. Data preparation and conversion is one of the most time-consuming and frustrating tasks related to speech recognition. For this reason, good data preparation tools are of utmost importance in speech recognition applications. A speaker independent system that uses HMM's, needs a corpus that is phonetically balanced. The recording must be done using people selected from a balanced demography of the population.

The two types of corpus - spontaneous-speech corpus and read-speech corpus can be identified

- Spontaneous-speech corpus is prepared by recording spontaneously spoken speech that is later transcribed by listening to the recorded speech.
- Read-speech corpus is prepared by creating a phonetically balanced prompt text, which

is later recorded by making a representative group of people read the text [12].

For a natural language, the preparation of a phonetically balanced prompt text can done by selecting available print text, including newspapers, fictional text, the Bible, etc. Once the text corpus is ready, the next step is recording the speech. For a speaker independent system the speech must be read by representatives from all demographic groups. It must be read by males and females of varying age groups. The selection of the people should also represent pronunciation differences of dialects of the language. This ensures that the final models are robust.

2.8. The HTK Toolkit

There are several toolkits that are designed to help the implementation of speech recognizers. These tools provide the basic and advanced facilities that a speech recognition system needs to build its models. Notable examples of these toolkits include, the MSSTATE Toolkit, the CSLU toolkit [18], [24] and the HTK toolkit [22]. Of these, the HTK toolkit has been chosen for this research thesis, for no other reason than its popularity among speech researchers specializing in Amharic. Since, there have been attempts to develop Amharic ASR using HTK, the results obtained can be judged against the results of the other researchers.

HTK is a toolkit for building Hidden Markov Models [27]. HTK can be used to model any general time series data. However, it was primarily designed to model HMM based speech signal processing functions (in particular recognizers). HTK, therefore, supports a large number of functions designed for handling speech-processing functions. It is composed of modules that have been designed to provide speech-processing functionalities. It is primarily a command line utility to allow researchers perform data preparation, training, testing and analysis. It also provides a graphic interface based utility to record and label speech sound files.

2.8.1. Data Preparation in HTK

If no readily available corpus exists then one of HTK's modules (HSLab) can be used to record the speech data directly from an audio device. The default file format is HTK native format. If the HSLAB utility available in the toolkit is used to create the speech corpus files, then the files will be generated using the HTK native format. HSLab is an interactive label editor for manipulating speech label files. HSLab can load a sampled waveform file, allow the user to determine the boundaries of the speech units of interest and assign labels to them. It can also

allow a user to load an already existing label file and edit the current label boundaries. HSLab has a primitive graphic interface with a modest functionality.

2.8.2. Feature Extraction in HTK

The recorded speech data must be parametrized into sequences of feature vectors. HTK supports both FFT-based and LPC-based analysis. Most commonly speech recognizers use the Mel frequency cepstral coefficients (MFCCs), which are derived from FFT-based on log spectra. The HCopy tool of HTK is used to convert speech data to parametric form. HTK lets the user select a wide variety of options for the parametrization including Mel frequency cepstral coefficients, linear prediction coefficients and linear prediction cepstral coefficients. These feature vectors can be extracted at any desired rate and the analysis frame varied as per the requirement of the application. Furthermore, various types of Windows can be used for the analysis frames including rectangular, Hamming and Hanning windows.

2.9. Acoustic Modeling in HTK

Acoustic models are built by training the HMM models using the parametrized speech data. In HTK, the training process takes place in stages. A context-dependent speech recognition that uses phonemes as the basic units needs to train models for each occurrence of triphones. To get to that point, however, it is common practice to start with a monophone system with a set of identical HMM parameters and single Gaussian output probability density. This initial model is called a flat start model. In this model every mean and variance is identical and is equal to the global mean of the speech corpus. These are then trained; short pause models are added and retrained. If there are phones that have multiple pronunciations, then the training data needs to be realigned, i.e. consider all possible pronunciation of the words in question and outputs the phone (pronunciation) sequence that best matches the acoustic data. Triphone transcriptions are then generated from the monophone transcription and state tying is performed to decrease the computational and training data requirement complexity. At each stage Baum-Welch retraining is done to arrive at the final set of models.

The tools HINIT and HEREST provide isolated word style training to create an initial set of models. For this purpose speech data with the monophone transcription and label must be provided. HINIT reads in all of the marked training data and isolates all occurrences of phones. It then iteratively computes an initial set of parameter values using a segmental k-mean procedure. On the first cycle, the training data is uniformly segmented, each model state is

matched with the corresponding data segments and then the means and variances are estimated. On the second and successive cycles, the segmental k-means procedure is replaced by Viterbi search to obtain more accurate results.

The initial parameter values computed by HINIT are then further re-estimated by HEREST. The fully labeled data is used to refine the estimation using the Baum-Welch algorithm. Normally, only a small subset of the training data is used to create the initial flat-start models. With this initial set of models, HEREST performs training using the entire set of data. HEREST is the core-training tool. At the end of its computation an estimate of HMM parameters for each model is obtained. This module is designed to handle large databases and can run on multiple computers to reduce computation time. Once decent parameter estimation is done the next step is to fix the silence modes. This is done by creating a transition between the last and first states of silence and by tying the only state of short pause to the middle state of silence. State tying is done by manually manipulating the files that represent the trained set of models. The system can then be retrained using the new triphone transcriptions and be ready for recognition.

2.10. Morphological Analysis

2.10.1. Basic Concepts

Morphology is the study of the way words are built up from smaller meaning bearing units i.e., morphemes [6]. A morpheme is often defined as the minimal meaning-bearing unit in language. So, for example, the word *fox* consists of a single morpheme (the morpheme fox) while the word *cats* consists of the morpheme cat and the morpheme –s. As this example suggests, it is often useful to distinguish two broad classes of morphemes: stems and affixes. The exact details of the distinction vary from language to language, but intuitively, the stem is the “main” morpheme of the word, supplying the main meaning, while the affixes add “additional” meanings of various kinds.

Affixes are further divided into *prefixes*, *suffixes*, *infixes*, and *circumfixes*. Prefixes precede the stem, suffixes follow the stem, circumfixes do both, and infixes are inserted inside the stem. Prefixes and suffixes are often called *concatenative morphemes* since a word is composed of as a number of morphemes concatenated together. A word can have more than one affix. For example, the word rewrites has the prefix re-, the stem write, and the suffix –s.

There are two broad (and practically overlapping) classes of ways to form words from

morphemes: *inflection* and *derivation*. Inflection is the combination of a word stem with grammatical morpheme, usually resulting in a word of the same class as the original stem, and usually filling some syntactic function like agreement. For example English has the inflection morpheme –s for making the plural on nouns, and the inflection morpheme –ed for making the past tense on verbs. Derivation is the combination of a word stem with grammatical morpheme, usually resulting in a word of a different class, often with a meaning hard to predict exactly. A very common kind of derivation in English is the formation of new nouns, often from verbs or adjectives. This process is called nominalization. For example the verb computerize can take the derivation suffix –ation to produce the noun computerization.

2.10.2. Amharic Morphology

Amharic, like other Semitic languages such as Arabic, exhibits the root-pattern morphological phenomenon. This is especially true of Amharic verbs. Amharic verbs consist of stems and affixes. A stem again consists of root consonants (typically three) and vowel patterns.

Stems encode different types of morphosyntactic information by changing the arrangement of the root consonants and vowel patterns [8] as shown in the sample derivational paradigm of the verb sbr (‘break’) in Table 2.1 [8].

Simple derived verb forms		
Categories	Pattern	Word
Perfect	CVCXVC[1]	*säbbär*
Imperfect	CVCC	*säbr*
Gerund	CVCC	*säbr*
Imperative	CCVC	*sbär*
Complex derived verb forms – Perfect aspect		
Causative	*as*-CVCXVC	*assäbbär*
Passive	*tä*-CVCXVC	*täsäbbär*
Reduplicative	CVCVCXVC	*säbabbär*

Table 2.1: Simple verb forms of sbr ("break").

(Note: The symbols C, V and X in the template forms indicate consonant, vowel, and germination of the previous consonant respectively.)

The derivational processes in the complex derived forms are either internal in which CV patterns

are changed, or external where derivational affixes (such as as-,tä-) are attached to the simple derived forms. Derivational processes may also involve a combination of internal and external changes. Some of the derivations in this class serve to express adverbial functions as the language has limited lexicalized adverbs. The stem forms take various affixes. For example, arguments of a verb are indicated on the verb using suffix pronouns (see Table 2.2). The subject is marked on the verb using subject suffix pronouns, which agree with the subject in number, gender, and person. The direct object and some prepositional phrase complements are optionally marked on the verb. Functional elements like negation maker, conjunctives (some of them), some auxiliary verbs and relative markers are also bound morphemes and are attached to the verb.

Person	Pronoun	Subject	Object	Posses.
Singular				
1st	*ənɛ*	*-ku/hu*	*-ñ*	*-ɛ*
2nd m	*antä*	*-k/h*	*-h*	*-h*
Pol.	*ərswo*	*-u*	*-wot*	*-ačč̈äw*
f.	*ančɨ*	*-š*	*-š*	*-š*
3rd m.	*əssu*	*-ä*	*-w*	*-u*
f.	*əsswa*	*-äč*	*-at*	*-wa*
Pol.	*ərsacäw*	*-u*	*ačč̈äw*	*ačč̈äw*
Plural				
1st	*əññə*	*-n*	*-ačč̈əhu*	*-ačč̈ən*
2nd	*ənnantä*	*-ačč̈əhu*	*-ačč̈əhu*	*-ačč̈əhu*
3rd	*ənnässu*	*-u*	*-ačč̈äw*	*-ačč̈äw*

Table 2.2: Independent and suffix pronouns of Amharic.

All arguments of a verb are optional and may only be indicated by suffix pronouns, that is, a verb may stand alone as a sentence.

Amharic nouns inflect for gender, number and case. However, gender in Amharic is mostly natural. Pre-nominal modifiers like adjectives, relative clauses take a similar set of inflectional morphemes as nouns. The head noun appears at the end of a noun phrase. Amharic has both preposition and postposition. Some of the prepositions are bound morphemes and are prefixed to nouns. Amharic also uses circumpositions to indicate positional relations. Table 2.3 shows sample prepositions and postpositions.

Preposit.	Meaning	Postposit.	Meaning
bä-	at, in	*lay*	Top
lä-	for	*wəst*	Inside
kä-	from	*fit*	Front
səlä-	about	$h^{w}ala$	back
əndä-	like		
wädä	to		

Table 2.3: Sample prepositions and postpositions in Amharic

The definite article in Amharic is a bound morpheme and is attached to a noun or to the first inflected element in a noun phrase whereas other determiners like demonstrative pronouns are independent morphemes.

In general, although the above description of Amharic morphology is far from complete, it is hoped that it gives some idea of its complexity. The distribution of the different affixal elements across different part of speech categories is varied. Some are specific to a particular part of speech category while others are applied across several parts of speech category.

2.11. Prior Works on Amharic ASR and Morphology

Syllable-Based Speech Recognition for Amharic a prior work by Solomon Teferra Abate and Wolfgang Menzel in Hamburg, Germany used the more or less a one to one correspondence Amharic orthography with syllabic sounds and develop a Consonant, Vowel (CV) syllable-based speech recognizer, using Hidden Markov Modeling (HMM), and achieved an encouraging word recognition accuracy (90.43%). Although there are still possibilities of performance improvement, they conclude that the use of CV syllables is a promising alternative in the development of ASRs for Amharic [9].

Pioneering the work on morphological analysis of Amharic verbs, Abiyot (Bayou, 2000) designed and implemented a prototype word parser for Amharic verbs and their derivation. He designed a knowledge-based system that parses verbs, and nouns derived from verbs. He used root pattern and affixes to determine the lexical and inflectional category of the words. He tested his system on a limited number of words (200 verbs and 200 nouns) and the result showed that

86% of the verbs and 84% of the nouns were recognized correctly.

Another prototype morphological analyzer for Amharic was developed by Tesfaye Bayu (Bayu, 2002) where he used an unsupervised learning approach based on probabilistic models to extract morphemic components (prefix, stem and suffix) to construct a morphological dictionary. He also investigated an approach whereby he applied the principle of Auto segmental Phonology to identify morphemic component of a stem such as consonantal root, vocalic melodies and CV-templates. The first system was able to parse successfully 87% of words of the test data (433 of 500 words). This result corresponds to a precision of 95% and a recall of 90%. Tested with 255 stems, the second system identified the morphemic components of 241 (or 94% of the) stems correctly.

Fissaha and Haller [8] discuss the morphology of Amharic verbs in the context of Machine Translation and present an implemetation of a morphological analyzer for Amharic using Xerox Finite State Tools (XFST). The different classification schemes for Amharic verbs that have been forwarded are discussed followed by the implication such classifications have on the implementation strategy. They claim that morphological analysis for Amharic with XFST can handle most of the morphological phenomena except some derivation processes, which involve simultaneous application of both stem interdigitation and reduplication.

Saba and Gibbon (Amsalu and Gibbon, 2005) extend the XFST implementation of Amharic morphology to include all word categories. Testing with 1620 words text from an Amharic bible, they report recall levels of 94% for verbs, 85% for nouns, and 88% for adjectives while they report precisions of 94% for nouns, 81% for adjectives, 91% for adverbs, and 54% for verbs, at the above specified recall levels.

A more recent work that applies Conditional Random Fields to segment and part of speech tag Amharic words is done by Fissaha (Adafre, 2005). He reports an accuracy of 84% for the word segmentation. The work deals with bound morphemes of prepositions, conjunctions, relative markers, auxiliary verbs, negation marker and coordinate conjunction, but leaves out other bound morphemes such as definite article, agreement features such as gender and number, case markers, etc, and considers them to be part of the word. The best result (84%) is obtained by using character, morphological and lexical features.

There has been a work done by Alemayehu and Willet (Alemayehu and Willett, 2002), which

investigates the effectiveness of stemming in information retrieval for Amharic. They compare performance of word-based, stem-based, and root-based retrieval of 40 Amharic queries against 548 Amharic documents, and show better recall levels for stem and root based retrieval over word based, but they don't provide information on the precision of these experiments.

All the above mentioned works attempt to address the need to develop a morphological analyzer for Amharic, and show that there has been a great deal of effort put in the design and implementation of each system. Although that is the case, none of them are publicly available, and/or are limited in some way [8].

Chapter 3 Text Corpus Development

The text corpus development is a required part of language modeling. Whether stochastic or deterministic language models are used, it is essential that all the words to be recognized be listed and available. Furthermore, a pronunciation dictionary be constructed for each of these words. A pronunciation dictionary is simply a listing of the words together with a symbolic description of their pronunciations. For ASR systems, this depends on the selected recognition units. If the basic recognition units are phonemes, then the dictionary must also give the pronunciations using these phonemes.

For deterministic language models, the task grammar needs to be defined using the EBNF grammar definition language. This requires the classification of all the words into classes and a definition of the possible word sequence.

In both cases, the initial task is the collection of the words. In the Amharic case the collection of these words is difficult because each base word has a large number of morphological derivations. In this thesis the development of the dictionary and grammar are both automated. This is done by collecting the base words, creating the derived morphological derivations of these words using a morphological expansion program and finally generating the pronunciation dictionary and the task grammar-using dictionary and grammar definition programs.

For the purpose of this thesis, only sample base words are collected. These words are summarized in table 3.1. These words are classified into verbs, adjectives and nouns.

Verbs	Adjectives	Nouns
በላ	ገዘዝ	ሕዝብ
ፈላ	ትኩስ	በሶ
ጠላ	ትልቅ	ብሹን
ሞላ	ትንሽ	ዳቦ
ሳላ	የሚጣፍጥ	ገዘፈ
		ከበደ ሻደ
		ሙላ

Table 3.1: Words collected for the SST-LVASR corpus development

The thesis has limited the classes of word to three types. The task grammar used in the thesis has been defined as

```
( Start( [$adjective] $noun [$adjective] $noun $verb ) end )
```

The task that now remains is to generate the morphological expansions of these words and create the pronunciation dictionary and the word classification part of the grammar definition. In the next sections, a detailed explanation of these procedures is given.

3.1 The Problem of Morphological Expansion

In general the expansion of stem words into their morphological forms involves three processes

- Suffixation – the addition of suffixes to the stem word,
- Prefixation – the addition of prefixes to the stem word
- Infixation – the addition of infixes to the stem word

Thus, there are three types of additions: *suffixes*, *prefixes* and *infixes*. In linguistics the word components i.e. stem words, suffixes, prefixes and infixes are referred to as *morphemes*. Suffixes are added at the end of the stem word, prefixes at the beginning and infixes appear in

the middle of the stem word.

Consider the verb **በላ** (he ate). We can consider this word as the stem word and some examples of the possible morphologically expanded forms include **በላች** (she ate) **አስበላ** (he made (it) to be eaten) **አስበላች**(she made (it) to be eaten). In the first example only a suffix (**ች**) is added to the stem word, in the second only a prefix (**አስ**) while in the last example both a prefix (**አስ**) and a suffix (**ች**) are added on the stem word to complete the words.

But there are cases where simple suffixation and prefixation are not sufficient to model the morphological construction rules. Consider these examples **አባላ** (he helped eating) **በሉ** (They ate) **አባሉት** (They helped him eating) **አስበሉት** (They made him eat).

In the first case the first letter of the main word changes from **በ** to **ባ**, in the second case the last letter of the main word changes from **ላ** to **ሉ** and as the other examples show these processes can be augmented by further simple prefixation of suffixation processes. Amharic linguists model these process by considering the stem word to be **በል** instead of **በላ** in which case **በላ** itself is a derived word of **በል** constructed by the following augmentation process **በል** + **ኣ**. Notice that using this traditional model the above examples can be thought of as being produced through the following expansion processes.

አባላ = **አ**+ **ብ**+**ኣ**+**ል**+**ኣ**

በሉ = **በል**+**ኡ**

አባሉት = **አ**+ **ብ**+**ኣ**+**ል**+**ኡ**+**ት**

አስበሉት = **አስ**+ **በል**+**ኡ**+**ት**

Thus, in this case the suffixation, prefixation and infixation processes are sufficient to explain the morphologically expanded word forms. For example, in the example **አባላ** the morpheme **ኣ** appears as an infix in the stem word **በል** appearing between **ብ** and **ል**, notice, however, when an infix appears the preceding letter need also be modified in this example from **በ** to **ብ**. In a similar manner **አባሉት** is constructed by augmenting the suffix **ኡት** on the stem word **በል**. The difficulty with this model of morphological expansion is that the (sub)morphemes **ኣ** and **ኡ** do not appear in the written form of the whole word. Additionally, the stem words in this model are not generally valid words on their own and are not part of the Amharic vocabulary. In the case

of **በል**, the stem word actually is part of the Amharic vocabulary but a completely unrelated word from **በላ**.

በላ = he ate,

በል = say (used as a command for the masculine gender)

Furthermore, infixation requires modifying some of the letters of the main word to properly model the construction process, which apparently makes the infixation process, at best, a weak model for the morphological derivation process in Amharic.

The reason for using words that are not part of the Amharic vocabulary as stem words is to assure that all stem words end in consonants. The majority of the letters of the Amharic abugida represent a concatenation of a consonant and a vowel phoneme. Thus while English uses only one symbol to represent the phoneme ***/b/***, Amharic uses seven - **በ ቡ ቢ ባ ቤ ብ ቦ**. These symbols (except the sixth) represent a concatenation of the phoneme /***b***/ with all the existing vowels. In fact, the above symbols represent the bi-phoneme sounds (except the sixth) ***b+ax b+u b+ee b+aa b+ay b b+o*** respectively. Thus, in Amharic writing the vowels are never explicitly shown unless they are specifically needed in certain words (e.g. **ሰ<u>ዐ</u>ት፡ <u>አ</u>ዲስ <u>ኢ</u>ት፡ዮጵያ**). The underlined symbols represent the phonemes /***aa***/, /***aa***/ and /***ee***/ respectively.

In the morphological expansion rule of Amharic, however, the vowels of the word are sometimes modified to produce new words. To deal with such changes on the last letter of a word, linguists consider the stem word simply to be as the sequence of letters remaining after the last vowel is dropped.

The Amharic Alphabet system which is known as the Amharic Abugida, is composed of rows and columns instead of a linear list of letters as is the case for the English alphabet. Each row contains seven columns. The rows represent the basic phonemes while the columns represent the vowels appended to the consonants. Thus each row of the alphabet lists the concatenation of the particular phoneme with the six vowels with one column (the sixth) reserved for the basic phoneme without any concatenation. The order of the vowels in the row is consistent for all phonemes and is given by the following positional assignment.

Position	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Vowelax	u	ee	aa	ay	none	o	

Table 3.2: Vowels and their positional assignment in Amharic Abugida

While dropping the last vowel of a word to construct stem words neatly solves the problem of changing vowels during suffixation, this problem unfortunately does not occur only with the last letter of a word. In fact, in most infixation cases for Amharic the vowel part of a letter is modified, only this letter appears in the middle of a word instead of at the end. Applying the same technique of dropping the vowel in letters, however, is not possible for all letters of words to model infixation as this would, absurdly, make all Amharic stem words to be made of consonants. This would be too confusing and might sometimes render the remaining word to seem completely unrelated with the derived words (e.g . **ጠጣ** (he drank) will be represented by **ጥጥ** (cotton)).

An even more difficult problem with this model is the fact that in certain circumstances the morphological expansions are obtained by changing not only the vowels but also the consonants themselves, for example **በላ** becomes **ብዩ** when it takes the form of command for a feminine subject. The last letter of **በላ** is converted from **ላ/*la*/** to **ዩ/*yee*/**. In this case the model of suffixation, prefixation and infixation completely fails.

3.2 A New Approach to Morphological Derivation

While the above model might be the preferred way of explaining morphological expansion by linguists, as it explains the word construction process in a conceptually simple framework of infixation, prefixation and suffixation, an alternative model is proposed in this thesis to specify the rules of morphological construction. In this model, the stem word is considered as **በላ** instead of **በል** (more generally the masculine past tense form is taken as the stem word for verbs). As in the previous model this model produces words by suffixation and prefixation, but unlike the first model considers word modifications as column and row movements in the Amharic abugida.

The difference between the two models is that when a vowel appears as a (sub)morpheme as in

the case of ኡ in በሉ = በል+ኡ this model thinks of the modification as a process moving a letter from its current position in the Amharic abugida to a new position within the same row. Thus, whether the problem occurs with suffixation or vowel infixation, vowel modifications can always be considered as movements of letters within a particular row of the Amharic alphabet. The advantage of this model over the first one is that it is not necessary to construct out of vocabulary words or resort to the unnatural decomposition of words into English like separate consonant and vowel combinations. Furthermore, the conversion of consonants presents no problem as they can be modeled as row movements.

This particular property can be especially useful for computer modeling of the morphological derivation rules. A possible symbolic representation of the morphological rules can be summarized as follows.

Simple prefixation and suffixation can be modeled by a combination of symbols in three braces, the first bracket represents the prefix symbols (**PS**), the second the stem word (**SW**) and the third bracket the suffix symbols (**SS**). Thus such expansions can be expressed as

$$\{PS\}\{SW\}\{SS\} \tag{3.1}$$

Either of the prefix symbols and suffix symbols can be empty in which case there will be no corresponding augmentation. In most cases the suffixes and prefixes are simple combinations of letters, but sometimes, the suffixes and prefixes might be selected letters from the word itself. In that case they can be indicated by a number indicating the position of the letter within the word.

When a particular letter in the word changes, it can be modeled as row or column movement in the Amharic abugida. The representation can be done using the following syntax

$$L > R_m C_n \tag{3.2}$$

$$L < R_m C_n \tag{3.3}$$

This modification rule representation must either precede or follow the **SW** symbol within the second braces in Equation (3.1).

Multiple such modification rules can be specified and can appear simultaneously preceding and following the **SW** symbol.

- The symbol L represents the position within the word on which the modification applies, thus if the 1st letter is to be modified L will be set to 1.
- The symbols > or < are used to indicate whether the positional count for the letter to be modified is done from the beginning of the word (left to right) or the end of the word (right to left) respectively.
- R is an indicator symbol (flag) that tells that the character moves within a row to a new position (horizontal movement). If the character R does not appear that means there will be no horizontal movement of the letter.
- C is an indicator symbol (flag) that tells that the character moves within a column to a new position (vertical movement). If the character C does not appear that means there will be no vertical movement of the letter.
- m tells to which position within a row the letter is to be moved. m takes a single digit since the word can be moved to positions 1 -7 only. m must follow the R flag.
- n tells to which row position the letter should be moved. n takes double digits since the letter can be moved to positions 1 -33. n must follow the C flag..

An additional specification in the script is the case where a new letter must be inserted somewhere in the middle of the word, proper infixatioin. This can be handled by using the *I* flag, which indicates insertion of letters. The insertion flag *I* must be followed by a number *o* indicating where the symbol is to be inserted and the inserted symbol *IS*. If the letter to be inserted is selected from the words of the letter, this can be indicated by using a number indicating the position of the letter for *IS*. Like all position specification, this can be done from the left or the right and can be indicated by attaching the symbols > or < respectively. If the inserted symbol must undergo horizontal or vertical movements then they can be indicated in the same manner as equations (3.2) and (3.3). Insertion will be handled by the following two alternative formats.

$$I_0 > IS \qquad (3.4)$$

$$L_0 \lessdot LS \qquad (3.5)$$

An example of such specification would be

{አ} {1 R6 በላ 1 R3C24} {በት·} (3.6)

This command will move the first letter of **በላ** to row position 6 and the last letter to row position 3 and column position 26. Furthermore it adds the prefix **አ** and the suffix **በት·**. In Effect this simple command transforms the word **በላ** (he ate) to **አብየቡበት·** (let his food be eaten). This example clearly demonstrates that modeling the morphological expansion rules of Amharic as horizontal and vertical movements of letters in the Amharic Abugida allows the specification and generation of the derived words in a simple script similar to equation (3.6).

The power of this model is its ability to express and hence generate the morphological expansions of words by grouping them into several classes. The morphological expansion rules of Amharic are largely regular. A script written for one word will equally apply to many other words. To exemplify this claim, let us take the words **በላ** (he ate), **ጠጣ** (he drank) and **ገዛ** (he bought) and some morphological expansions of these words.

Script	**Role**			
{}{SW}{}	3rd person singular, masculine subject, past tense, no object, direct action	**በላ**	**ጠጣ**	**ገዛ**
{ }{SW}{ች}	3rd person singular, feminine subject, past tense, no object, direct action	**በላች**	**ጠጣች**	**ገዛች**
{**አስ**}{SW}{}	3rd person singular, masculine subject, past tense, no object, third person action	**አስበላ**	**አስጠጣ**	**አስገዛ**
{ }{SW}{**በት**·}	3rd person singular, masculine subject, past tense, third person masculine object owner, direct action	**በላበት·**	**ጠጣበት·**	**ገዛበት·**
{ **አስ**}{SW}{ች}	3rd person singular, feminine subject, past tense, no object, third person action	**አስበላች**	**አስጠጣች**	**አስገዛች**
{ }{SW}{***ሁት***·}	1st person singular subject, past tense, third person singular object, direct action	**በላሁት·**	**ጠጣሁት·**	**ገዛሁት·**
{ **አ**}{1>R4 SW}{**ችው**}	3rd person singular, feminine subject, past tense, no object, third person singular, masculine helper action	**አባላችው**	**አጣጣችው**	**አ,ጋዛችው**
{ **አ**}{SW}{<1}	3rd person singular, masculine subject, past tense, no object, helper repetitive action	**አበላላ**	**አጠጣጣ**	**አገዛዛ**

Table 3.3: Some examples of morphological derivations for the two letter words ***በላ, ጠጣ,*** *and* ***ገዛ*** *and their corresponding scripts*

Notice that a single script can be used for all three words to generate the morphological derivations of one particular role. The scripts indicated in table 3.3 are all for two letter words. Not all of these rules, however, will work for three or four letter words. Table 3.4 shows that inspection of the morphological rules are identical for three lettered words.

Script	**Role**			
{SW}	3rd person singular, masculine subject, past tense, no object, direct action	**ደመረ**	**መከረ**	**ቀጠፈ**
{ }{SW}{**ች**}	3rd person singular, feminine subject, past tense, no object, direct action	**ደመረች**	**መከረች**	**ቀጠፈች**
{**አስ**}{SW}{}	3rd person singular, masculine subject, past tense, no object, third person action	**አስደመረ**	**አስመከረ**	**አስቀጠፈ**
{ }{SW}{**በት·**}	3rd person singular, masculine subject, past tense, third person masculine object owner, direct action	**ደመረበት·**	**መከረበት·**	**ቀጠፈበት·**
{ **አስ**}{SW}{**ች**}	3rd person singular, feminine subject, past tense, no object, third person action	**አስደመረች**	**አስመከረች**	**አስቀጠፈች**
{ }{SW 1<R6}{ **ት·**}	1st person singular subject, past tense, third person singular object, direct action	**ደመርኩት·**	**መከርኩት·**	**ቀጠፍኩት·**
{ **አ**}{1>R4 SW}{**ችው**}	3rd person singular, feminine subject, past tense, no object, third person singular, masculine helper action	**አዳመረችው**	**አማከረችው**	**አቃጠፈችው**
{ **አ**}{I2>2>R4 SW}{}	3rd person singular, masculine subject, past tense, no object, helper repetitive action	**አደማመረ**	**አመካከረ**	**አቀጣጠፈ**

Table 3.4: Some examples of morphological derivations for the three letter words ***ደመረ, መከረ, and ቀጠፈ*** *and their corresponding scripts*

This suggests that a classification of the words must be made in terms of their morphological derivation rules. One obvious grouping is according to how many letters the word has. A script written for two letter words should handle all two-letter words and that for three words should handle all three words etc.

Even if we limit ourselves to words of the same length, this claim is not always true. There are some irregularities in the morphological rules, which force the algorithm to fail. An example would be the case of a no subject, no object, 3rd person singular feminine command action. This is given in table 3.5 for the two lettered words **በላ**, **ጠጣ**, **ገዛ**, **ቀረ** and **ሞተ**.

Word	Derived Form	Script
በላ	ብዩ	{ }{1>R6 SW 1<R3C24}{ }
ጠጣ	ጠጪ	{ }{1>R1 SW 1<R3C28 }{ }
ገዛ	ግገር	{ }{1>R6 SW 1<R3C25 }{ }
ቀረ	ቅሪ	{ }{1>R6 SW 1<R3 }{ }
ሞተ	መቺ	{ }{1>R2 SW 1<R3C12 }{ }

Table 3.5: Morphological derivations for the words **በላ**, **ጠጣ**, **ገዛ** *and* **ሞተ** *and their corresponding scripts for the no subject, no object, 3rd person singular feminine command action role.*

As table 3.5 clearly demonstrates, these derived forms cannot be generated by a single script. It shows that for all four words the column movements are different. Furthermore, you will notice that even the horizontal movements of the first letters in the words are different. In the cases of **በላ**, **ገዛ**and **ቀረ** the row movement is to position 6, while for **ጠጣ** it is to position 1 and for **ሞተ** it is to position 2.

The reason for the column discrepancies is a result of the consonant phonemes that appear at the end of the stem words. As a general rule no column movements are necessary in any morphological derivation. However, if the last consonant in the word is an alveolar (**የድድ**) phoneme, and the row movement for the last letter is to position 3 (as is the case in the above example), the consonant must be converted to its corresponding postalveolar (**የ(ድ)ላንቃ**) counterpart. The phonemes **ል**, **ጥ**, **ዝ** and **ት** are all alveolar sounds; hence they are converted to **ይ**, **ጭ**, **ሸ**and **ች** respectively. On the other hand is not an alveolar sound hence there is no column movement for the word **ቀረ**.

Alveolar (የድድ) Consonant	Postalveolar (የ(ድ)ላንቃ) counterpart
ል	ይ
ስ	ሽ
ት	ች
ጥ	ጭ
ዝ	ዥ
ድ	ጅ
ን	ኝ

Table 3.6: Amharic alveolar phonemes and their postalveolar counterparts

Since the rules of column movements are known, the morphological expansion script can be redesigned to automatically detect alveolar sounds and convert them to post alveolar sounds as per table 3.6.

The discrepancies in row movements are a little more complicated than the column movement discrepancies. They are related to the etymology of the words and cannot be inferred from the arrangement or inherent properties of the letters alone. In this particular example, the default rule for conversion is to move the first letter of the word to the sixth position. However, we notice that this fails for words like ሞተ. ሞተ (he died) is derived from the ge'ez word መወተand has inherited its morphological rule. The second letter /*w*/ is a semivowel and Amharic has converted the concatenation of መ /*m*/ and ወ /*w*/ to ሞ /*mo*/. Thus in the morphological expansion ሞተ must be considered as a three lettered word. The rule will then be መውቺ { }{2>R6 SW 1<R3C12 }{ }. Combining the መው sounds, we arrive at መቺ, which is the correct derivation.

The above examples clearly demonstrate that the proper derivation of morphological forms is a difficult task that requires the proper classification and expansion rule of words. Some of the classifications that must be made include

- Basic word type: verb, noun, adjective etc.
- Transitive verb vs. non-transitive verb.
  - **መታ** (he hit) derives **መታው** (he hit him) but **ሞተ** (he died) does not derive **ሞተው** (he died him)
- Verbs that end with ***ax*** vs. verbs that end with ***aa***
  - **መታ** derives to **እስኪመታ** but **ቀረ** derives to **እስኪቀር**
- Verbs that start with consonants vs. verbs that start with vowels
  - **ቀረ** derives to **እስኪቀር** but **አሸ** derives to **እስኪያሸ**
- Nouns taken from ge'ez vs. native Amharic nouns
  - **ሰው** derives to **ሰዎች** but **ጳጳስ** derives to **ጳጳሳት**

The research done in this thesis with respect to the morphological properties of Amharic words has shown great promise, however, a complete specification of the morphological rules is beyond its scope. In this research, the derivation of the words with respect to their morphological rules has been performed by limiting the rules to those that do not pose complexities similar to the ones discussed in the preceding paragraphs.

3.3 Implementation of the Morphological Expander

The morphological expander program is an implementation of the column and row movement model of morphological derivation. It has the ability to parse scripts similar to equation (3.6) and take a list of base words. Depending on the script, the number and variety of derivations generated by this program can vary. It has the ability to read scripts from a text file, view and inspect the expansions as applied to a particular stem word and save the results in another text file. The user interface of the program is shown in Figure 3.1. The current implementation of this program has not considered automatic checking and classification of words. Furthermore, a complete study of all possible classifications has not been made. Neither insertions, nor specifications of morphemes by repeating existing letters is not handled in the current version. The following list summarizes the features supported by the current implementation of the

morphological expander.

- Allows the addition of prefixes and suffixes,
- Allows specification of horizontal movements,
- Allows specification of vertical movements,

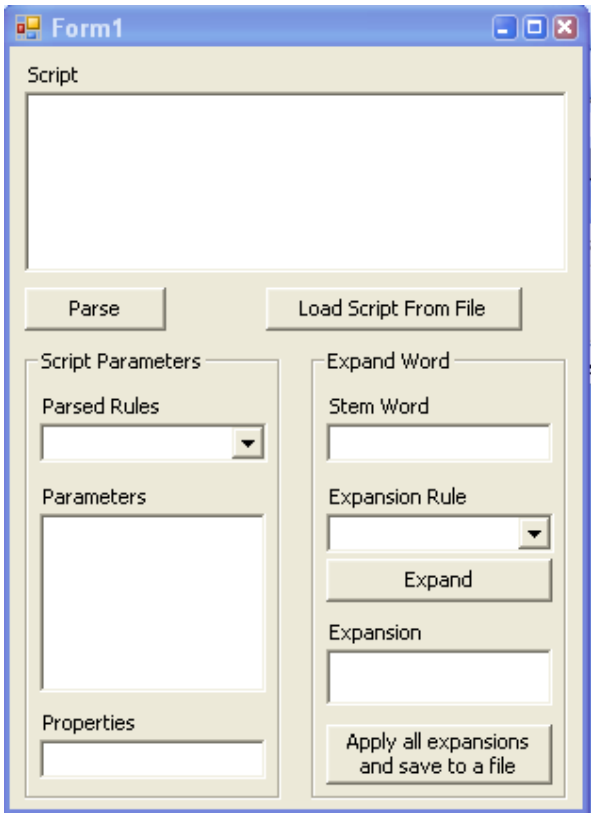

Figure 3.1: The user interface of the morphological expansion program

3.4 Performance and Result of the Morphological Expander Program

A script dealing with 349 different morphological derivations for two letter words was written and parsed by the morphological expander. These derivational forms were chosen to apply specifically to transitive words starting with consonants. Column movements were not applied as they need an implementation that considers the type of consonant at the end of the word. These and other aforementioned conditions are not technically difficult to handle, however, they are time consuming as it requires a thorough study and specification of all possible classes of

words and the rules of derivational expansions. This script was tested with the four transitive two lettered words በላ ጠጣ ሰፋ and ሰራ. The result indicates that the expansion accuracy was 100%.

3.5 The Generation of the Dictionary and the Task Grammar

The dictionary is simply a list of words and their corresponding phonetic pronunciation labels. The dictionary development is made for Amharic because Amharic graphemes have an easy mapping to their pronunciations. Each grapheme has a distinct sequence of phones. The dictionary development thus requires the expansion of the graphemes into these sequences of phonemes. The mapping between the graphemes and the phonemes is already given in figure 1.1. A program is then designed to go through the list of words and make these expansions.

The word classification of the grammar development is an even easier task. The output of the morphological expander has already classified the words as either adjectives, verbs or nouns. These words must now be reformatted as per the EBNF syntax – which basically means inserting the logical or operator "|" between the words. Another program is designed to handle this task to complete the text corpus development.

Chapter 4 Speech Corpus Development

A speech corpus, as far as ASR systems are concerned, is simply a set of recorded and labeled sentences. The speech corpus is required during the acoustic model development, where the Baum-Welch algorithm is used to train the HMM models. The sentences that make up the speech corpus are selected so that they cover the words that appear in the language, furthermore, sufficient samples for each of the phonemes appearing in the language must be provided. Normally these sentences are recorded and labeled manually using speech labeling tools like the HSLAB tool of HTK toolkit, or the WaveSurfer utility. In this thesis, however, the speech corpus is developed by concatenation of the morphemes of pre-recorded words.

The starting point in acoustic model development is the preparation of sound samples of all the phones that appear in the system as they appear in the context of sentences. These sound samples are then parametrized (features extracted) using one of the available feature extraction algorithms. The sound samples to be used in the model development must be labeled to indicate the time durations over which the phones (the realization of the phonemes) appear.

The training prompt sentences are simply a list of valid sentences for the language model of the task. Since, the system uses a deterministic language model as described in figure 5.1, the training prompts are simply sentences that obey this rule. These sentences can quickly be generated by the inbuilt HTK tool HSGen, which simply takes the specified grammar file as an input and generates the required number of sentences. In this thesis, 100 such training sentences were generated. The complete list of the training sentences is given in Annex I-F. The total word count of these sentences is 380.

The sentences so generated must now be recorded and the appearance of each phone labeled indicating the start and end time value of each of the phones. This step is normally accomplished by recording the sentences using human speakers and manually time labeling the appearance of each of the phonemes. This, however, is accomplished differently in this thesis. The recording and labeling is done only for a selected set of isolated words (kernel words). Segments of these words are then programmatically concatenated to generate the rest of the words (non-kernel words) in the vocabulary. The labels of these new sets of words are also generated programmatically by appropriately selecting the durations of the phonemes from the labels of the component sound files of the words. At this point sound samples of all of the words in the

vocabulary are available. These words are then concatenated one after the other according to the list of words in each entry of the training prompt sentences. The labels for the whole sentence are also generated from the labels of the component words. These process flows is shown in figure 4.1.

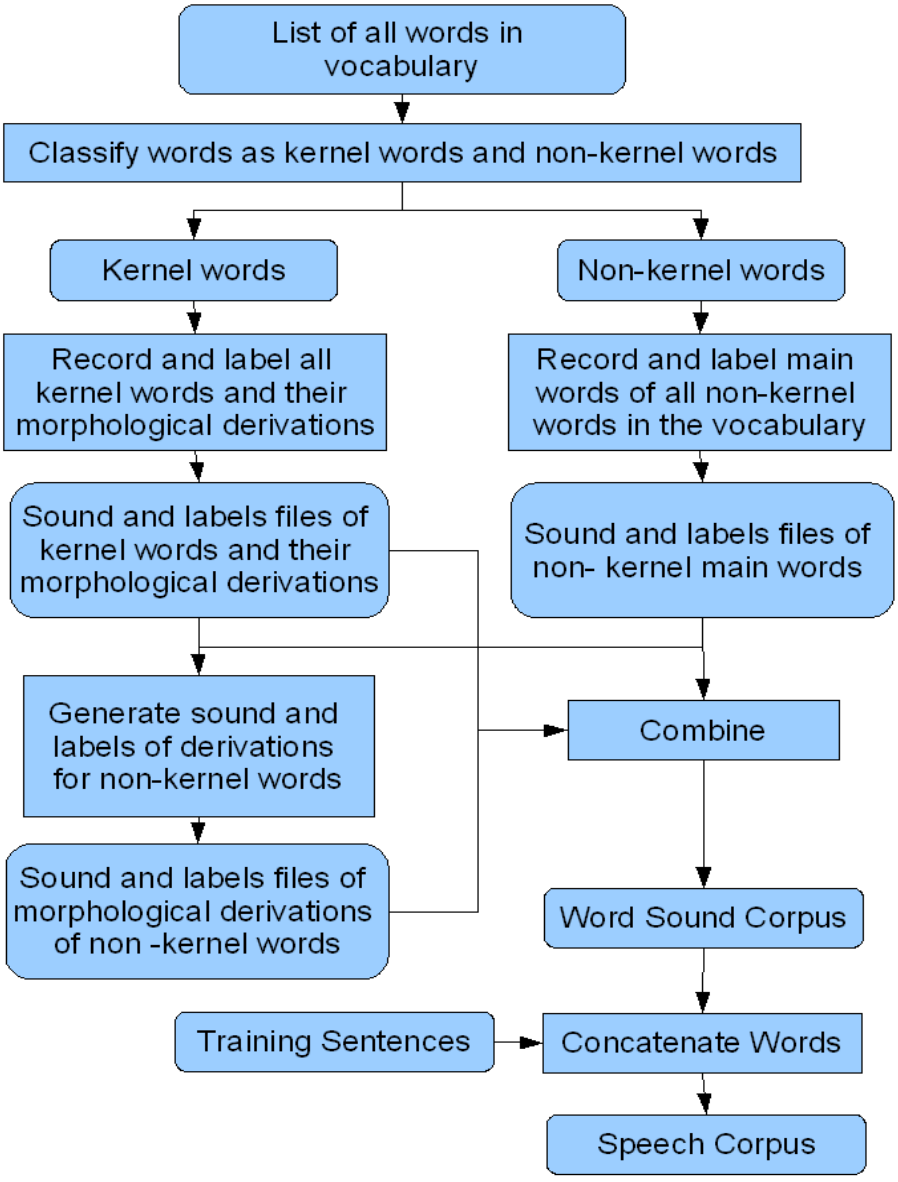

Figure 4.1: Speech corpus development process

4.1 Selection of the Kernel Words and Recording the Base Data

The kernel words are those for which all derivations are recorded. These words are selected in such a way that all the prefixes and suffixes that appear in the non-kernel words are covered; furthermore the last and first phonemes of the kernel words must cover the last and first phonemes of the non-kernel words. This last requirement is to assure that the concatenation of the morphemes can be done by cutting the first and last phonemes of the kernel words together with the preceding and succeeding morphemes of the prefixes and suffixes respectively, thus reducing the effect of co-articulation. Since, the morphemes are dependent on the different word classes, this needs to be done for each of the word classes (and subclasses) in the vocabulary.

Inspection of table 4.1 reveals that for the case of nouns the pairs **ከበደ** and **አበበ**, **በሶ** and **ዳቦ** end with the same phoneme /***ax***/ and /***o***/ respectively while **ብሉ᎓ን**, **ሻይ**, **ገበሬ** and **ውሃ** have distinct end phonemes. As for the starting phoneme only **በሶ** and **ብሉ᎓ን** have a common phoneme /***b***/, the rest being distinct. In the case of verbs, all words end with the same phoneme /***a***/ while the start phoneme is distinct for all of the verbs. For the case of adjectives, none of them have identical end phoneme, but the three words **ት·ኩ·ስ**, **ት·ል·ቅ**, **ት·ን·ሽ**all have /***t***/ as their start phoneme. Furthermore, the adjectives have only suffixed derivational forms. Thus the procedure can only put one of the verbs as a kernel word generating all the suffixes in the problem. The gain obtained by using the nouns as kernel words is minimal and there is no gain at all by using the adjectives as kernel words.

	Verbs		Nouns		Adjectives	
	Recorded	**Generated**	**Recorded**	**Generated**	**Recorded**	**Generated**
1.	በላ		አበበ		ጎበዝ	
2.	አበላ		አበበን		ጎበዙ	ጎበዙን
3.	አስበላ		ከበደ	ከበደን	ትኩስ	ትኩሱን
4.	አስበላው		በሶ		ትኩሱ	
5.	በላበት፡		በሶውን		ትልቅ	ትልቁን
6.	በላው		ዳቦ	ዳቦውን	ትልቁ	
7.	ፈላ	አስፈላው	ብሎን		ትንሽ	ትንሹን
8.	አፈላ	ፈላበት፡	ብሎኑን		ትንሹ	
9.	አስፈላ	ፈላው	ሻይ	ሻዩን		
10.	ላላ	አስላላው	ገበሬ			
11.	አላላ	ላላበት፡	ገበሬውን			
12.	አስላላ	ላላው	ውሃ			
13.	ሞላ	አስሞላው	ውሃውን			
14.	አሞላ	ሞላበት፡				
15.	አስሞላ	ሞላው				
16.	ጠላ	አስጠላው				
17.	አጠላ	ጠላበት፡				
18.	አስጠላ	ጠላው				

Table 4.1: List of manually recorded and generated words

Therefore, of the 58 words in the vocabulary, 39 will be manually recorded and labeled while 19 will be automatically generated and labeled. Multiple samples of the 29 words must be available for robust training. In this thesis, 5 different samples of each of the 29 words have been recorded and labeled. The total word count of the prompt sentences is 380. This means that the total number of words that must be recorded and labeled is equal to 380. Using the generation of the sentences by concatenation, the total required recording and labeling is just 145. Thus, even with this very limited experimental setup, the recording and labeling effort is only 51% of that required by the normal procedure. This is indeed a significant improvement, but is expected to increase even more significantly if the number of words and the derivational forms is increased.

The recording and labeling of the kernel words and their derivations can be performed by the inbuilt HTK tool HSLAB or other speech processing programs like wavesurfer. These tools provide a graphical user interface for recording, listening, viewing, editing and assigning of labels. Figure 4.2 shows the labeling of the word ሰላ, using the HSLAB tool.

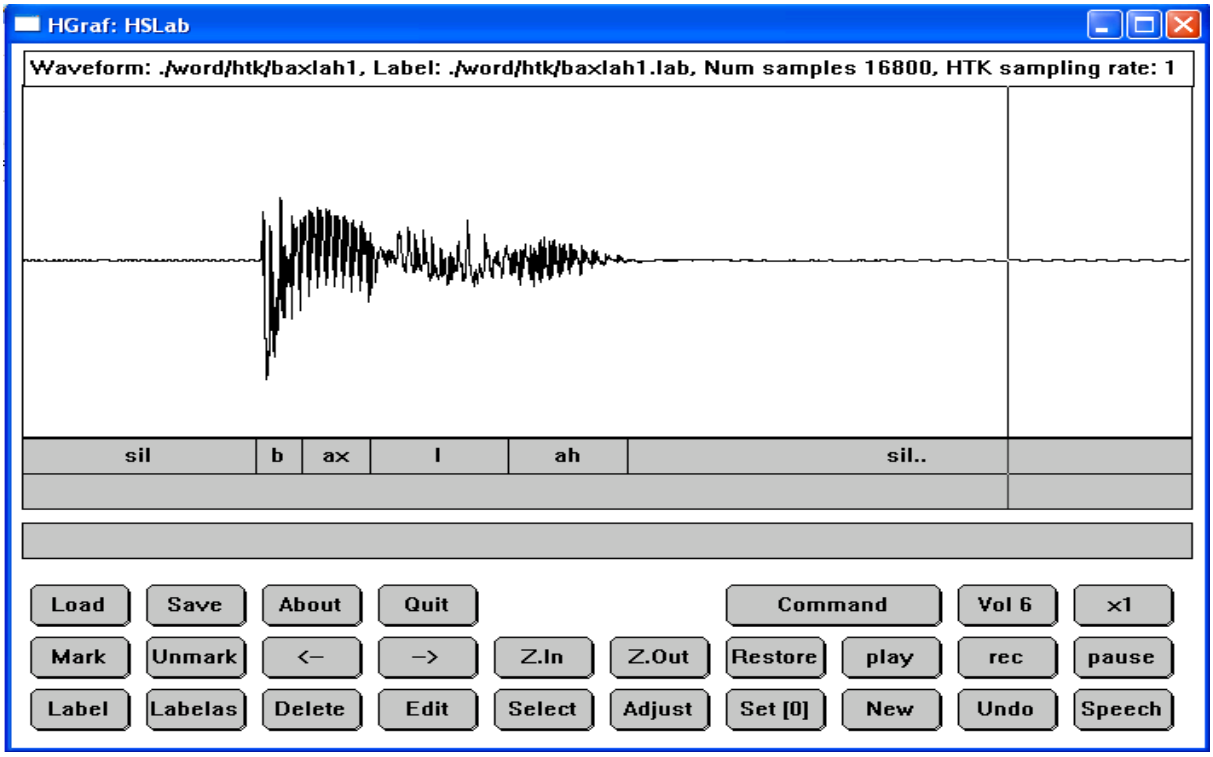

Figure 4.2: Monophone labeling in HSLAB for the word ሰላ

4.2 Generation of the Word Speech Corpus

Once the base data (the kernel words, their derivation and the stem words of the non-kernel words) sound waveforms and their corresponding labels are available, the next task is to generate the sound waveforms and corresponding labels of the non-kernel derivations. The key idea is to cut segments of speech from the recorded sound files of the kernel derivations and the base word and concatenate them appropriately. The concatenations are based using the basic principle of cutting the edge phones of the two sound forms in the middle and appending them to each other. For example, to generate the waveform for ፊላበት = /***f ax l ah b ax t***/, the sound files for ፊላ = /***f ax l ah***/ and በላበት= /***b ax l ah b ax t***/ are used. The segment of speeches to be concatenated are /***f ax l ah***/ and /***b ax t***/. However, instead of simply taking /***b ax t***/ from /***b ax l ah b ax t***/, we take the speech segment starting from the middle of the last ***ah*** all the way to **t**. We also take the speech segment from /***f ax l ah***/, that starts from f ends at the middle of ***ah.*** This will reduce the effect of co-articulation. The labels can also be generated from the labels of the two words. The label for ፊላበት will be identical to the label of ፊላ from the start up to the phoneme ***l.*** The duration of the edge phone ***ah*** is calculated by adding half of the duration for the ***ah*** that appears in /***f ax l ah***/ and half of the duration of ***ah*** that appears in /***b ax l ah b ax t***/ the durations of the phones /***b ax t***/ are identical to their corresponding phones in /***b ax l ah b ax t***/. The absolute locations are computed by adding these durations to the absolute time value of each of the previous phones. Since there are multiple samples of both ፊላ and በላበት, there will also be multiple samples of the newly generated word ፊላበትobtained by creating all possible combinations of the morphemes. Since there are exactly 5 samples of each, the new word will have 25 different samples. A matlab program (expandword.m) that performs this concatenation has been written and by applying this program on all words, the word speech corpus is generated. The word speech corpus is simply a set of multiple sound and label files for all the words in the vocabulary.

4.3 Generating the Speech Corpus

The next step is the generation of the sound and corresponding label files for the sentences in the

training prompt. This is the final speech corpus to be used in the training of the HMM models. The requirement here is to concatenate the sound files of the words in the sentences and generate both the sound and label files. Unlike the previous step, no intra-word concatenation occurs. Only the sequence of words must properly appear. Two issues to consider are the selection of one of the sound files (as there are multiple samples for each word), and the modeling of the inter-word silence – the so called short pause. The first can be handled by a random selection of the recorded word samples. In the second case, a simple algorithm of taking one tenth of the total silence duration is taken. The reason behind selecting this value is the empirical result that inter-word pause was found to be one tenth of the sentence end pause. This corpus generation is performed by the matlab program (sentgen.m).

4.4 Coding the Data (Feature Extraction)

The last step in the data preparation stage is to parametrize the speech waveform files into sequences of feature vectors. HTK supports both FFT-based and LPC-based analysis. Most commonly speech recognizers use Mel frequency cepstral coefficients (MFCCs), which are derived from FFT-based log spectra. The HCopy tool is used to convert speech data to parametric form. HTK lets the user select a wide variety of options for the parametrization. The user has the option of choosing several settings for the parametrization, including:

- Target kind: e.g., MFCC, linear prediction coefficients, linear prediction cepstra,
- Target rate: rate of feature vector extraction,
- Window type: rectangular, hamming,
- Window size,
- Pre-emphasization threshold.

Chapter 5 Implementing the SST-LVASR System

As was mentioned in section 1.7.1, the objective of this thesis is to investigate the possible use of machine synthesized speech based on morphological rules in the development of a large vocabulary continuous Amharic speech recognition system. The tasks involved in the thesis are development of the text corpus, development of the speech corpus, training a speech recognizer system using the HTK toolkit using synthesized speech and testing the results. In chapter 3 and 4, the development of the text corpus and speech corpus were discussed respectively. In this chapter, the procedure of training the recognizer and the testing procedures and results are described.

5.1 Experimental Setup

The SST-LVASR Amharic speech recognizer system developed in this experiment is a large vocabulary speaker specific continuous system. This means that the number of distinct words are large, the speech to be recognized is from a single person and the speech need not have artificial pauses between words.

5.1.1 Language Modeling

The language modeling for this thesis uses a deterministic language model. This means that the language is specified by defining the grammar of the language. The grammar definition needs identifying the words, grouping them into word classes and specifying allowed word sequences. Furthermore, a pronunciation dictionary is required as detailed in section 3.5.

5.1.1.1. Generating the Collection of Words (The Text Corpus)

As described in chapter 4, the words that are to be recognized by the system are generated in a two-step process. In the first step the set of basic words to be recognized are collected. In the second step these words are expanded using the morphological expander program to generate the derived forms of these words. In this thesis the type of words to be collected has been limited to adjectives, nouns and verbs. Therefore, the base words to be collected must fall into one of these three groups. A script must then be written to these three classes (and subclasses) of these words so that their derived forms can be expanded by the morphological expander. Table 5.1 shows the basic words used in the thesis.

Nouns	Verbs	Adjectives
አበበ	በላ	ጎበዝ
በሶ	ፈላ	ትኩስ
ብሎን	ላላ	ትልቅ
ዳቦ	ሞላ	ትንሽ
ገበሬ	ጠላ	
ከበደ		
ሻይ		
ውሃ		

Table 5.1: Basic words used in the experiment

A script is then used for each of these words to generate selected derivational forms of these words. In the first attempt 349 different forms for verbs, 2 forms for nouns and 3 forms for the adjectives were generated. This resulted in 1745 different verb forms, 16 different forms for nouns and 12 different forms for adjectives for a total of 1773 different words in total. By simply increasing the number of basic words this total number can be increased significantly. For example if we use 100 verbs, 100 nouns and 100 adjectives, this number would increase to 35400 words in total.

As will become clearer in the next section, the recording and labeling requirements for 1773 words is quite large, therefore another script was also used to generate a different set of word derivations with the objective of limiting the number of words within a single class of words. This script generates only six different verb forms, two noun forms and three adjective forms. In this case the total number of words to be identified were only 58. This list of words is given in table 4.1.

5.1.1.2. The Pronunciation Dictionary

The pronunciation dictionary is simply a text file listing all the words in the system followed by the sequence of phonemes that define the word. The important decision to make here is which recognition unit to use. Possible candidates are phonemes and syllables. In this thesis, phonemes are used as the basic unit of recognition. Therefore, the task of generating the pronunciation dictionary involves breaking down the orthographic symbols into their component phonemes. Furthermore, at this point it is necessary to represent the Amharic orthographic symbols using English character symbols. This last requirement is to facilitate the handling of Amharic written text using characters accessible to computer programs.

	Verbs	Nouns	Adjectives
1.	በላ	አበበ	ጎበዝ
2.	አበላ	አበበን	ጎበዙ
3.	አስበላ	በሶ	ጎበዙን
4.	አስበላው	በሶውን	ትኩስ
5.	በላበት	ብሎን	ትኩሱ
6.	በላው	ብሎኑን	ትኩሱን
7.	ፈላ	ዳቦ	ትልቅ
8.	አፈላ	ዳቦውን	ትልቁ
9.	አስፈላ	ገበሬ	ትልቁን
10.	አስፈላው	ገበሬውን	ትንሽ
11.	ፈላበት	ከበደ	ትንሹ
12.	ፈላው	ከበደን	ትንሹን
13.	ላላ	ሻይ	
14.	አላላ	ሻዩን	
15.	አስላላ	ውሃ	
16.	አስላላው	ውሃውን	
17.	ላላበት		
18.	ላላው		
19.	ሞላ		
20.	አሞላ		
21.	አስሞላ		

22.	አስሞላው		
23.	ሞላበት		
24.	ሞላው		
25.	ጠላ		
26.	አጠላ		
27. 28.	አስጠላ አስጠላው		
29.	ጠላበት		
30.	ጠላው		

Table 5.2: Words generated using the morphological expander program

Fortunately, for Amharic, there is a simple mapping of the orthographic symbols and their pronunciations. This is because Amharic graphemes represent a consonant vowel concatenation. Thus a table need be constructed that maps the Amharic vowels and consonants to symbolic phonemes. For the phonemes that appear in this thesis, this mapping is given in table 5.3.

	Amharic Phoneme	**Symbolic Representation**
1.	አ	ah
2.	እ	ax
3.	ብ	b
4.	ን	n
5.	ል	l
6.	ስ	s
7.	ው	w
8.	ት	t
9.	ኦ	o
10.	ኡ	u
11.	ድ	d
12.	ፍ	f
13.	ግ	g
14.	ር	r
15.	ኤ	e
16.	ዝ	z
17.	ክ	k

18.	ም	m
19.	ሽ	sz
20.	ይ	y
21.	ቅ	q
22.	ጥ	tz
23.	ህ	h
24.	ኢ	ee

Table 5.3: Amharic phonemes of the ASR system and their symbolic representations

With this symbolic mapping, the representation of the words in English characters and the generation of the dictionary become a straightforward task. In this thesis, the words are represented by concatenation of the English phonemic characters. The pronunciation is generated simply by replacing each grapheme with a concatenation of the symbolic phoneme sequences representing the word. A sample entry from the dictionary using this rule will look like.

ahbaxbax	ah b ax b ax
ahbaxbaxn	ah b ax b ax n
ahbaxlah	ah b ax l ah
ahsbaxlah	ah s b ax l ah
ahsbaxlahw baxlah	ah s b ax l ah w ah b ax l ah
baxlahbaxt	b ax l ah b ax t
baxlahw	ah s b ax l ah w

Table 5.4 Sample entries of the pronunciation dictionary

This dictionary is automatically generated by using a matlab program (dict.m) specifically designed to perform this task. The complete dictionary file so generated is given in Annex -A. The lists of all phonemes that appear in the task are given in Annex -B.

ASR systems based on triphone modeling also require that a separate dictionary with triphone definition, be available. This just means that every appearance of a phoneme (monophone) must consider its right and left contexts. Thus, considering the word **ahbaxbax**, the triphone pronunciation will be **ah+b ah-b+ax b-ax+b ax-b+ax b-ax.** Note that, in this definition, each phoneme appearance is augmented with its previous and next phoneme with a – and a + sign respectively to indicate that the phoneme appears sandwiched between these two phonemes. This description is called a triphone modeling. Note that if the phoneme appears at the beginning or end of a word, it is described as a biphone. A sample triphone pronunciation dictionary entry is given in table 5.5.

ahbaxbax	ah+b ah-b+ax b-ax+b ax-b+ax b-ax
ahbaxbaxn	ah+b ah-b+ax b-ax+b ax-b+ax b-ax+n ax-n
ahbaxlah	ah+b ah-b+ax b-ax+l ax-l+ah l-ah
ahsbaxlah	ah+s ah-s+b s-b+ax b-ax+l ax-l+ah l-ah
ahsbaxlahw	ah+s ah-s+b s-b+ax b-ax+l ax-l+ah l-ah+w ah-w
baxlah	ah+b ah-b+ax b-ax+l ax-l+ah l-ah sp
baxlahbaxt	b+ax b-ax+l ax-l+ah l-ah+b ah-b+ax b-ax+t ax-t
baxlahw	ah+s ah-s+b s-b+ax b-ax+l ax-l+ah l-ah+w ah-w

Table 5.5 Sample entries of the triphone pronunciation dictionary

The triphone pronunciation dictionary can be generated from the monophone dictionary by looking at the right and left contexts of a phoneme in the word. This is performed by the inbuilt HTK tool HLED. The complete triphone dictionary is given in Annex -C. The corresponding list of triphones is given in Annex -D.

5.1.1.3. The Task Grammar

The language model of the ASR system in this thesis uses a very simple deterministic syntax

```
(start-sent ([$adjective] $noun [$adjective] $noun $verb) end-sent)
```

Figure 5.1: EBNF specification of the task grammar used in the thesis.

which is composed of verbs, nouns and adjectives. This grammar syntax is shown in Figure 5.1.

In this grammar three types of words are used – nouns, adjectives and verbs. An acceptable sentence can be composed of sequences of an optional adjective, a noun, an optional adjective, a noun and a verb. The key words start-sent and end-sent indicate the start and end of a sentence. To complete the grammar definition, all that is required is to identify the set of adjectives, nouns and verbs. The output of the morphological expander program has classified the words into their proper classes, thus all that is required is to reformat the words to fit the formatting requirement of EBNF as described in section (2.6.1). The matlab program dict.m is able to perform this task in addition to generating the dictionary. The grammar file so generated is given in Annex -E.

5.1.2 Acoustic Modeling

Acoustic modeling using HMM means the development of HMM models for each recognition unit. Since, this thesis uses triphone modeling, the HMM models must be calculated for each triphone that appears in the words. The main algorithm applied to get to these models is the Baum-Welch retraining algorithm. This algorithm requires that a starting model average be available, using which, increasingly accurate model parameters are recalculated.

The development of the final models is performed in several stages, in each step more accurate values for the model parameters being computed. These stages can be broken down into three main stages – data preparation (speech corpus), monophone modeling and tied state triphone modeling. In the data preparation stage, the speech corpus to be used as the training data set for the development of the models is constructed. This has already been described in chapter 4.

In the monophone modeling stage, the speech corpus is used to train the monophone models that appear in the words of the language. Finally, the monophone models are extended and refined to

develop tied state triphone models.

5.2 Monophone Model Development

The monophone models are trained values of the parameters of HMM, trained using Baum-Welch for each monophones that appear in the vocabulary. The training of these models takes several steps. In the first step a global average for the HMM values is calculated to give starting model parameters for the Baum-Welch retraining algorithm. These models are then further refined to get more accurate values for the models. A special step is taken to tie the short pause model that appears between words with one state of sentence end silence model. This is required because, the samples available for short pause are insufficient to calculate accurate model parameters. In our case, this tied state modeling has a further advantage of removing errors introduced by the concatenation of words. These procedures are performed in the same manner as other ASR system development systems.

5.3 Tied-State Triphone Model Development

Context-dependent triphones can be made by simply cloning monophones and then re-estimating using triphone transcriptions. This is accomplished in three stages; the first of these simply clones the existing monophones to triphone models. For example, if the monophone /***ah***/ occurs in 5 contexts, five models are created for it, but each of them will have the same model parameters.

In the second stage, these new models can now be refined by Baum-Welch re-estimation. In this case, the triphone labels for the same speech corpus is used.

At this stage, we have a set of trained tri-phone models. However, the number of samples for the triphones is usually small and hence the resulting model parameters are not accurate. This data sparsity problem can be reduced by tying the states of similar triphones. Tied-state triphones are generated, by grouping together all those triphones which are acoustically similar. However, the choice of which states to tie requires a bit more subtlety since the performance of the recognizer depends crucially on how accurate the state output distributions capture the statistics of the speech data. Two mechanisms can be used which allow states to be clustered and then each cluster tied.

The first is data-driven and uses a similarity measure between states. The second uses decision trees and is based on asking questions about the left and right contexts of each triphone. The decision tree attempts to find those contexts which make the largest difference to the acoustics and which should therefore distinguish clusters. In HTK, triphone state tying is performed by the inbuilt tool HHED. In this thesis, the decision tree method has been employed. An edit script which contains the instructions regarding which contexts to examine for possible clustering must be created. This script must classify the different phonemes according to their type. The classification of the phonemes is done according to their manner of articulation, whether they are consonants or vowels, whether they are voiced or not and their duration. The script is rather long and requires proper study of the phonemes that appear in the vocabulary. Once this script is available, the decision tree is constructed by HHED and the states of the triphones can be tied according to the output of the decision tree. Final retraining of these tied state triphones completes the construction of the ASR system, which is now ready for recognition and testing.

Chapter 6 Recognizer Evaluation and Analysis of Results

The recognizer is now ready to accept speech inputs and recognize sentences. An imperative restriction of the system is that the sentences must be composed of only the words defined in the dictionary of the recognition system and that their sequence must obey the grammar for the task. With this restriction, the recognizer should be able to identify the said sequences of words. Such recognizers are bound to make errors and the question now is how big are the number of errors made during recognition. Two tests were performed to test the accuracy of the system. The first of these tests was performed using speech synthesized in the same manner as the training speech corpus. The second test was performed using naturally recorded sentences. In both of these tests a test speech corpus with 50 sentences were used. The test prompt sentences were generated using the HGen inbuilt tool in much the same way as that of the training speech corpus. Two sets of speech corpora were then generated. The first of these were generated using the sentgen.m program by concatenation of morphemes again in the same manner as the training speech corpus. The second speech corpus was generated by manually recording and labeling, these same sentences. These labeled and parametrized speech samples are then recognized using the HVITE recognition utility of HTK. The results for the two tests are summarized in tables 6.1 and 6.2.

	% Correct	**Accuracy**	**Correct**	**Deletion**	**Substitutions**	**Insertions**	**Total**
Sentences	68.00%	-	34	-	16	-	50
Words	90.32%	90.32%	168	1	17	0	186

Table 6.1: Test results for recognition accuracy using synthesized speech

	% Correct	Accuracy	Correct	Deletion	Substitutions	Insertions	Total
Sentences	20.00%	-	10	-	40	-	50
Words	62.37%	60.75%	116	3	67	3	186

Table 6.2: Test results for recognition accuracy using natural speech

6.1 Results Analysis

Table 6.1 shows that when speech synthesized in the same manner as that of the training speech corpus is used for testing, out of the 50 sentences 34 (68%) were correctly identified and out of a total of 186 words 168 (90.32%) were correctly identified. This is, indeed, a relatively accurate result considering that only 5 samples of each of the base words have been used for training. Furthermore, because of the randomization step in selecting the sound samples of the morphemes, the sound samples used in the recognition are different from that of those used in the training. Thus, under this limitation, the 90% accuracy achieved at this stage can be considered quite accurate – indicating that training of the models in the HTK toolkit was performed accurately.

The objective of the thesis, of course, is to see if synthetic speech trained ASR system can be used to recognize natural speech. Thus, the main result of the investigation is summarized in the test using naturally recorded speech. The big question is, does the system have good accuracy in recognizing naturally uttered speech. As table 6.2 shows the sentence correct percentage is 20.00%, and the word accuracy is 62.37%. This is indeed a significant result. It indicates that using synthetic speech for training at least 62% of the words is correctly identified. In other words, out of the 186 words 116 were correctly identified by the system. These numbers suggest that with synthetic speech some level of recognition is possible, giving the impetus for more research in finding ways to increase this accuracy.

Remembering that some of the speech samples for the words are not produced by concatenations, it is instructive to separate the word accuracy percentages according to whether the words speech samples were produced by concatenation or not. Table (6.3) gives the comparative percentage accuracy of words separated by whether their training samples were produced by concatenation or not.

Words	% Correct	Accuracy	Correct	Deletion	Substitutions	Insertions	Total
Concatenated	51.52%	51.52%	17	0	16	0	33
Unconcatenated	64.71%	62.75%	99	3	51	3	153
Overall	62.37%	60.75%	116	3	67	3	186

Table 6.3: Test results for recognition accuracy using natural speech classified by whether the training speech samples of the words were generated using concatenation or not.

This result is a good way of identifying the origin of recognition error. Notice that the recognition accuracy for the concatenated words alone is 51%, which is about 9 percentage points lower than the global word recognition accuracy and 11 percentage points lower than the accuracy of the unconcatenated words alone. As expected concatenation resulted in lower accuracy levels, but not by much. The origins of the error include the following.

- the 62.75% accuracy for the unconcatenated words indicate that the recording of the words individually instead of as they appear in context has reduced the recognition accuracy.
- The 9-percentage points reduction in the accuracy of the concatenated words indicate that the concatenation itself has introduced some error. This however is not as big a source of error as that of the recording of the words out of context.

Chapter 7 Concluding Remarks

The objective of this thesis was to investigate the possible use of machine synthesized speech based on morphological rules in the development of a large vocabulary continuous Amharic speech recognition system. The end result expected from this research was an answer to the question of whether the development effort of large vocabulary speech recognizers can be significantly reduced by using machine synthesized speech based on morphological rules. To arrive at the answer the development process of the ASR system was decomposed into three stages.

- Development of the text corpus aided with a morphological expansion program to generate the derived forms of base words,
- Development of the speech corpus using an automatic speech synthesis program based on pre-recorded speech samples of the base words and their morphemes,
- Implementation of the ASR system using the HTK toolkit.

The reduction expected from this new way of speech synthesizer development is to be achieved by reducing the effort of the text and speech corpora development, which are arguably the two most time taking and expensive stages of ASR development. The results section of this document has already described that this algorithm is encouraging and might quicken the development of a general purpose Amharic ASR system comparatively easily.

As part of the development of the SST-LVASR system, a morphological expander program has been implemented. Incidentally this program is a powerful program with potentially very useful applications. Hence, a special section is reserved in this chapter to remark about this program and its potential applications.

7.1 Remarks about Text Corpus Development

The text corpus for the ASR system is developed using the morphological expander program, which was designed to accept the rules of derivation of Amharic base words and generate all the morphological derivations. The program models morphological derivations as row and column movements on the Amharic Abugida. This model gave promising results in its ability to generate the derivations. For large types of derivations, the expander program can take a single

script and generate hundreds of derivations for large classes of words. However, certain derivations were found to have irregularities, which were difficult to handle using the program. These irregularities arise from different reasons, some of which are the location of certain phonemes on certain positions of the word, the origin of the word, the type of word etc.

While the generation of the derived forms for any type of derivation can be modeled by this program, the main difficulty is in designing a generic rule that can automatically apply the appropriate rules for each type of word. This is a formidable problem and requires a thorough understanding of the morphological expansion rules of all types of words. This requires a study be made to classify words according to their etymology, phoneme position, type and all other factors affecting their morphological derivation. With this information available, scripts can be designed which can accurately and completely generate the derived forms of words and hence a largely complete text corpus.

Even with its limitations, however, the program can still serve as a great tool in the generation of text corpus. This is because most morphological derivations are regular and relatively simple scripts, like the one used in the thesis, can still generate accurate derivations for the words. This can be combined with a manual checking stage to correct inaccuracies and fill in omissions. Although no experiments have been performed how many accurate results are obtained for highly varied classes of words (as this was not the objective of the research), the rather limited experiments performed in this thesis are indicative of the potentially significant reduction in text corpus development through this process.

7.2 Applications of the Morphological Expander Program

The applications of the morphological expander program are not necessary limited to a text corpus development. It can be integrated with other applications to serve a much more diverse purpose than its initial intended objective. The following list indicates some of the potential applications for the morphological expander program.

- **Advanced Amharic Searching:** searching algorithms for Amharic can be designed as exact match searches. This, however, is rather simplistic and might not always give the desired results. Internet searches, for example, return not only the word being searched but also the morphologically related words, as a Google search on the word math will demonstrate. Not only does the search return pages having the word math but also the words maths, mathematics, mathematical etc. Before such a search can be made the

derived forms of the word must be known. The program is designed exactly to provide these derived forms, hence its great potential in searching.

- **Amharic Spell Checking and Correction:** spell checking in word processors, for example, requires the availability of the words with proper spelling. Therefore, the development of a complete Amharic text corpus is essential to design a spell checking and correction program. Although the text corpus, which is the end result of the expander program, can be used as a database for the spell checker, the expander program with proper rules for all the words serves better than the static text corpus database. This is because, the number of words in the Amharic database is extremely large and the searching of the text corpus every time a spelling must be checked will result in unacceptable delays and memory requirement. If, on the other hand, the program is used to generate the words on the fly, both the delay and memory requirements can be significantly reduced. Therefore, the morphological expander program can be used to provide the necessary word database efficiently (in terms of both searching delays and storage space).

- **Advanced Amharic Sorting and Indexing:** sorting can be done by simply looking at the words individually and arranging them alphabetically. However, in certain applications, it may be desirable to collect derived words under their main word instead of their simple alphabetical positions. This for example is true in dictionaries and key word indexes at the end of written documents. Like in the previous two applications to achieve this objective, it is necessary to identify the base words with their morphological forms. Sorting and indexing programs can be integrated with the morphological expander for this advanced operation.

- **Automatic Translators:** many languages have automatic translation programs, which can convert written documents in one language into another. This operation essentially requires the availability of all the words of both languages and their corresponding meanings. The morphological expander helps this operation by producing all the words of Amharic. One possible way of proceeding is to use the text corpus so generated and find the corresponding meaning map. However, with modification, the expander program can be made to automatically generate these meaning maps as well. Since the morphologically derived word are generated using scripts and each derivation attaches certain predefined meaning changes, the program can be customized to generate these

meanings as well, provided that the meaning of the base word is given and the meaning changes associated with each derivation is associated with its corresponding script entry. Like the case of searching, these meaning maps are generated on the fly and hence the program is an efficient implementation – requiring less delay and storage space.

- **Application in Text-To-Speech Synthesis (TTS):** For speech synthesis, the text corpus is essential. But in addition, it requires a pronunciation dictionary, which is not necessary identical to the pronunciation dictionary of ASR systems. This pronunciation dictionary must contain phoneme duration, prosodic tagging and other information required by the synthesis engine. The expander program can help this text corpus development. In the first place, it can provide the basic list of words. Secondly, it might be possible to define the pronunciation modifications of each derivational forms based on the pronunciations of the base word and the participating morphemes. If this is the case, then, like the ASR speech corpus, the TTS text corpus can be fully automated. Lastly, like the cases for searching, sorting and indexing, this pronunciation entry can be generated on the fly to reduce the computational cost of searching a static TTS dictionary.

  A further application in TTS systems can be the breaking down of words in to their component morphemes. Many state of the art speech synthesizers allow morpheme concatenation in addition to diphone concatenation during speech synthesis, e.g., the Festival speech synthesis system. If the morphological decomposition of the words is given to these systems, they can use morpheme concatenation to produce more natural synthesized speech. Since the program has the ability to decomposed words into their component morphemes, this can greatly improve the naturalness of the synthesized speech. If in addition, the speech corpus is fully developed, the speech synthesizer can be made fully to be on morphemes based as the basic synthesis unit, even more greatly improving the quality of the synthesized speech.

- **Application in Speech-to-Sign Language Converters:** The ASR system developed in this thesis is designed to recognize each word in the vocabulary of the language. If, however, a system is required to convert speech to sign language, for the sake of people with hearing disability, this direct mapping is not desirable. In sign language, many of the morphological derivations of words are lumped together as one sign, possibly an additional sign appended to indicate the particular action. Therefore, such systems require the association of the morphologically derived word with the sign corresponding

to the base word and a possible association of the morphemes (or more exactly the action represented by the morphemes) with a further sign. This requires the decomposition of the identified word into the base word and the morphemes, which can be done using the morphological expander program. One way of handling this is creating a database of signs and their associated words, which can be based on the text corpus. This, however, can be implemented efficiently by decomposing the words on the fly, again using the morphological expander program.

7.3 Remarks about the Results of the SST-LVASR

To reduce the requirement of recording and labeling of a large number spoken sentence, the thesis envisaged the usage of synthetic speech based on the concatenation of morphemes. The speech corpus was developed by recording all the base words in the vocabulary and the base words and derived forms of a select group of words, which cover the diphone and morpheme requirements of all the words in the vocabulary. This corpus was then used to train an ASR system which is to be used to recognize naturally spoken speech. No literature which claimed the use of synthetic speech corpus in ASR training exists, as far as known, this is a new concept. The results analysis has shown that, this type of corpus development might be useful in speech recognizer development, as a 60.75% overall accuracy has been obtained with a 51.52% for the concatenated words. This accuracy is still low by commercial standards. Even so, improvements in the algorithm used for the collection of the initial kernel word database, the concatenation algorithm, as well as using larger training database might prove that this accuracy level can be increased to acceptable levels. Furthermore, the synthetic speech database can be used as a starting model development and be combined with conventional (naturally recorded) speech corpus to refine the models and get improved accuracy. In any case, the reduction in labeling and recording is still very high, proving that synthetic speech indeed is useful for ASR development

7.4 Recommendations on Future Work for SST-LVASR

The SST-LVASR is specifically designed to reduce the effort (and cost) of developing a general purpose ASR system for Amharic. As such, its main application is the reduction of the effort of development of general-purpose Amharic ASR systems to manageable levels. The procedures followed in this thesis can be mimicked on a larger set of words with improvements to arrive at a complete general purpose Amharic ASR system with a fraction of the time and cost compared

with conventional methods. Some of the improvements recommended are

- Application of more refined algorithms on the morpheme concatenation algorithm to accurately mimic natural speech. These may involve
    - application of filtering to remove concatenation spikes,
    - prosodic modification to model
        - duration of phonemes, syllables and morphemes,
        - remove beginning and end of word changes in pitch,
        - normalize loudness in the utterances.
- Using larger number of recordings for each kernel word to provide more samples of the speech,
- Using the kernel words in context (as parts of sentences), instead of recording them individually to remove the effect of longer durations, higher amplitudes and rises and falls of pitch when words are uttered individually,
- Combining conventional training methods with synthetic speech training: the recognition accuracy obtained in this research is not large enough for commercial applications, however, this accuracy is expected to increase by applying the previous four recommendations. If the accuracy obtained with those recommendations is still not high enough, the method can be combined with conventional methods to improve the accuracy, albeit with lesser data requirements. This can be approached in three different ways, which may be done independently or together
    - Designing an auxiliary natural speech corpus and using this corpus in conjunction with the synthetic speech corpus,
    - Completing the ASR development as in this thesis, conducting tests to identify words with low recognition accuracy and finally creating a naturally recorded speech corpus to retrain the models,
    - Designing an adaptation speech corpus of naturally recorded speech and applying it at the end of the development process.

Annex– Files Used in the Thesis

Annex – A The Dictionary File

```
ahbaxbax     ah b ax b ax
ahbaxbaxn    ah b ax b ax n
ahbaxlah     ah b ax l ah
ahsbaxlah    ah s b ax l ah
ahsbaxlahw   ah s b ax l ah w
baxlah       ah b ax l ah
baxlahbaxt   b ax l ah b ax t
baxlahw      ah s b ax l ah w
baxso        b ax s o
baxsown      b ax s o w n
blon         b l o n
blonun       b l o n u n
dahbo        d ah b o
dahbown      d ah b o w n
faxlah       f ax l ah
faxlahbaxt   f ax l ah b ax t
faxlahw      f ax l ah w
gaxbaxre     g ax b ax r e
gaxbaxrewn   g ax b ax r e w n
gobaxz       g o b ax z
gobaxzu      g o b ax z u
gobaxzun     g o b ax z u n
kaxbaxdax    k ax b ax d ax
kaxbaxdaxn   k ax b ax d ax n
lahlah       l ah l ah
lahlahbaxt   l ah l ah b ax t
lahlahw      l ah l ah w
molah        m o l ah
molahbaxt    m o l ah b ax t
molahw       m o l ah w
```

szahy	sz ah y
szahyun	sz ah y u n
tkus	t k u s
tlq	t l q
tlqu	t l q u
tlqun	t l q u n
tnsz	t n sz
tnszu	t n sz u
tnszun	t n sz u n
tzaxlah	tz ax l ah
tzaxlahbaxt	tz ax l ah b ax t
tzaxlahw	tz ax l ah w
whah	w h ah
whahwn	w h ah w n

Annex – B List of Monophones

ah

ax

b

n

l

s

w

t

o

u

d

f

g

r

e

z

k

m

sz

y

q

tz

h

ee

sil

Annex– C The Triphone Pronunciation Dictionary

ahbaxbax	ah+b ah-b+ax b-ax+b ax-b+ax b-ax
ahbaxbaxn	ah+b ah-b+ax b-ax+b ax-b+ax b-ax+n ax-n
ahbaxlah	ah+b ah-b+ax b-ax+l ax-l+ah l-ah
ahsbaxlah	ah+s ah-s+b s-b+ax b-ax+l ax-l+ah l-ah
ahsbaxlahw	ah+s ah-s+b s-b+ax b-ax+l ax-l+ah l-ah+w ah-w
baxlah	ah+b ah-b+ax b-ax+l ax-l+ah l-ah
baxlahbaxt	b+ax b-ax+l ax-l+ah l-ah+b ah-b+ax b-ax+t ax-t
baxlahw	ah+s ah-s+b s-b+ax b-ax+l ax-l+ah l-ah+w ah-w
baxso	b+ax b-ax+s ax-s+o s-o
baxsown	b+ax b-ax+s ax-s+o s-o+w o-w+n w-n
blon	b+l b-l+o l-o+n o-n
blonun	b+l b-l+o l-o+n o-n+u n-u+n u-n
dahbo	d+ah d-ah+b ah-b+o b-o
dahbown	d+ah d-ah+b ah-b+o b-o+w o-w+n w-n
faxlah	f+ax f-ax+l ax-l+ah l-ah
faxlahbaxt	f+ax f-ax+l ax-l+ah l-ah+b ah-b+ax b-ax+t ax-t
faxlahw	f+ax f-ax+l ax-l+ah l-ah+w ah-w
gaxbaxre	g+ax g-ax+b ax-b+ax b-ax+r ax-r+e r-e
gaxbaxrewn	g+ax g-ax+b ax-b+ax b-ax+r ax-r+e r-e+w e-w+n w-n
gobaxz	g+o g-o+b o-b+ax b-ax+z ax-z
gobaxzu	g+o g-o+b o-b+ax b-ax+z ax-z+u z-u
gobaxzun	g+o g-o+b o-b+ax b-ax+z ax-z+u z-u+n u-n
kaxbaxdax	k+ax k-ax+b ax-b+ax b-ax+d ax-d+ax d-ax
kaxbaxdaxn	k+ax k-ax+b ax-b+ax b-ax+d ax-d+ax d-ax+n ax-n
lahlah	l+ah l-ah+l ah-l+ah l-ah
lahlahbaxt	l+ah l-ah+l ah-l+ah l-ah+b ah-b+ax b-ax+t ax-t
lahlahw	l+ah l-ah+l ah-l+ah l-ah+w ah-w
molah	m+o m-o+l o-l+ah l-ah
molahbaxt	m+o m-o+l o-l+ah l-ah+b ah-b+ax b-ax+t ax-t
molahw	m+o m-o+l o-l+ah l-ah+w ah-w
szahy	sz+ah sz-ah+y ah-y

szahyun	sz+ah sz-ah+y ah-y+u y-u+n u-n
tkus	t+k t-k+u k-u+s u-s
tlq	t+l t-l+q l-q
tlqu	t+l t-l+q l-q+u q-u
tlqun	t+l t-l+q l-q+u q-u+n u-n
tnsz	t+n t-n+sz n-sz
tnszu	t+n t-n+sz n-sz+u sz-u
tnszun	t+n t-n+sz n-sz+u sz-u+n u-n
tzaxlah	tz+ax tz-ax+l ax-l+ah l-ah
tzaxlahbaxt	tz+ax tz-ax+l ax-l+ah l-ah+b ah-b+ax b-ax+t ax-t
tzaxlahw	tz+ax tz-ax+l ax-l+ah l-ah+w ah-w
whah	w+h w-h+ah h-ah
whahwn	w+h w-h+ah h-ah+w ah-w+n w-n

Annex – D List of Triphones

```
sil
g+o
g-o+b
o-b+ax
b-ax+z
ax-z
sp
k+ax
k-ax+b
ax-b+ax
b-ax+d
ax-d+ax
d-ax+n
ax-n
w+h
w-h+ah
h-ah+w
ah-w+n
w-n
l+ah
l-ah+l
ah-l+ah
l-ah+b
ah-b+ax
b-ax+t
ax-t
b+ax
b-ax+s
ax-s+o
s-o
sz+ah
```

sz-ah+y

ah-y

f+ax

f-ax+l

ax-l+ah

l-ah

ah+b

b-ax+b

b-ax

d+ah

d-ah+b

ah-b+o

b-o+w

o-w+n

tz+ax

tz-ax+l

g+ax

g-ax+b

b-ax+r

ax-r+e

r-e+w

e-w+n

b-o

ah+s

ah-s+b

s-b+ax

b-ax+l

l-ah+w

ah-w

d-ax

t+n

t-n+sz

n-sz+u

sz-u+n

u-n

b-ax+n

sz-u

b+l

b-l+o

l-o+n

o-n+u

n-u+n

r-e

ah-y+u

y-u+n

h-ah

t+l

t-l+q

l-q

m+o

m-o+l

o-l+ah

ax-z+u

z-u+n

z-u

l-q+u

q-u

s-o+w

n-sz

o-n

q-u+n

y+ax

y-ax+m

ax-m+ee

m-ee+tz

ee-tz+ah

```
tz-ah+f
ah-f+tz
f-tz
t+k
t-k+u
k-u+s
u-s
```

Annex – E The Task Grammar

```
$adjective = gobaxz |gobaxzu |gobaxzun |tkus |tlq |tlqu |tlqun |tnsz |tnszu
|tnszun;

$verb = ahbaxlah |ahsbaxlah |ahsbaxlahw |baxlah |baxlahbaxt |baxlahw |faxlah
|faxlahbaxt |faxlahw |lahlah |lahlahbaxt |lahlahw |molah |molahbaxt |molahw
|tzaxlah |tzaxlahbaxt |tzaxlahw;

$noun = ahbaxbax |ahbaxbaxn |baxso |baxsown |blon |blonun |dahbo |dahbown
|gaxbaxre |gaxbaxrewn |kaxbaxdax |kaxbaxdaxn |szahy |szahyun |whah |whahwn;

( sent-start( [$adjective] $noun [$adjective] $noun $verb ) sent-end )
```

Annex – F The Training Prompt Sentences

1. gobaxz szahy szahyun lahlahbaxt
2. blonun yaxmeetzahftz szahyun faxlahw
3. szahy gaxbaxrewn ahsbaxlah
4. szahy baxsown lahlahbaxt
5. tkus dahbown tlqu blonun ahbaxlah
6. dahbo blon ahbaxlah
7. ahbaxbaxn ahbaxbax ahsbaxlahw
8. gaxbaxrewn whah lahlahw
9. ahbaxbaxn blon lahlahw
10. whahwn tlqun whah molah
11. whah kaxbaxdax faxlahw
12. gaxbaxrewn gaxbaxre faxlahw
13. whahwn ahbaxbaxn ahbaxlah
14. gaxbaxre tnszun whah molahw
15. whah tlqun gaxbaxre tzaxlahw
16. whah yaxmeetzahftz dahbown tzaxlahw
17. baxso baxsown ahsbaxlahw
18. whahwn blon molah
19. tnszun dahbo blon lahlah
20. kaxbaxdax gobaxzu blon lahlahw
21. tnszun whah whah faxlah
22. tlqu baxso ahbaxbax molahbaxt
23. blon gobaxzu ahbaxbax ahbaxlah
24. gaxbaxre szahy lahlahw
25. gobaxzun baxsown dahbown molahbaxt
26. blonun gaxbaxrewn faxlah
27. whah tlqun baxso faxlah
28. gobaxzu kaxbaxdax ahbaxbax baxlahw
29. gaxbaxre tlq baxso tzaxlah
30. tlq gaxbaxrewn gobaxzu blonun ahsbaxlahw
31. baxso tnsz ahbaxbax baxlahbaxt

32. blonun tlqu szahyun baxlah
33. baxso dahbown lahlahw
34. szahy tkus dahbo baxlah
35. baxsown blon baxlahw
36. yaxmeetzahftz baxso gobaxzu dahbown faxlah
37. dahbown blon molahbaxt
38. szahyun tnsz blonun faxlahw
39. tnszun blon baxsown faxlah
40. gobaxzun kaxbaxdaxn dahbo baxlah
41. baxsown tlq blon ahsbaxlah
42. ahbaxbaxn tnszu baxso baxlahw
43. szahyun tlqu blon ahsbaxlahw
44. dahbo kaxbaxdaxn baxlahw
45. baxsown gaxbaxrewn molah
46. dahbo gobaxz whah tzaxlahbaxt
47. gobaxzu gaxbaxrewn tnszu baxso ahsbaxlahw
48. tlqun ahbaxbaxn gobaxzu kaxbaxdax lahlah
49. baxso whahwn faxlahbaxt
50. kaxbaxdaxn tlqun whah faxlahw
51. ahbaxbaxn whahwn molahw
52. tnszun baxso gaxbaxre faxlah
53. kaxbaxdax gaxbaxrewn ahsbaxlahw
54. tkus gaxbaxre kaxbaxdaxn faxlah
55. tlq szahy gobaxzu ahbaxbaxn tzaxlahw
56. szahy baxsown lahlah
57. tnszu szahyun kaxbaxdax faxlah
58. whahwn gobaxzun dahbo molah
59. szahy blonun faxlah
60. whahwn szahyun tzaxlahbaxt
61. tkus whah yaxmeetzahftz gaxbaxrewn ahsbaxlah
62. tkus baxsown dahbo baxlahw
63. gobaxzun dahbo blon faxlahbaxt
64. kaxbaxdaxn whahwn lahlah

65. gobaxzu blon szahyun baxlah
66. tlq szahyun ahbaxbax tzaxlahbaxt
67. baxsown gobaxzu szahy tzaxlahbaxt
68. tnszu dahbo tlqu dahbo tzaxlah
69. kaxbaxdax tkus kaxbaxdax tzaxlahbaxt
70. ahbaxbax tlqu gaxbaxre molah
71. dahbo baxso ahsbaxlah
72. blon blon ahsbaxlahw
73. tnsz kaxbaxdax kaxbaxdaxn lahlahbaxt
74. baxsown baxsown lahlahw
75. ahbaxbax dahbo ahbaxlah
76. whahwn kaxbaxdax faxlahw
77. tkus whah tkus kaxbaxdax baxlahw
78. dahbown gobaxz kaxbaxdax lahlahbaxt
79. kaxbaxdax tlqu blon faxlahw
80. whahwn dahbown tzaxlah
81. tnszun dahbo ahbaxbax faxlahw
82. ahbaxbaxn tlqu gaxbaxrewn faxlahw
83. tnszun dahbo gobaxzu blonun tzaxlah
84. kaxbaxdax tnsz baxsown faxlahbaxt
85. gobaxz baxso dahbo molah
86. tnszun ahbaxbax gobaxzu szahy tzaxlahbaxt
87. tlq ahbaxbax tkus kaxbaxdax lahlahw
88. tlqu ahbaxbaxn ahbaxbax molah
89. gaxbaxrewn dahbown tzaxlahbaxt
90. blonun tnszun szahy ahbaxlah
91. tnszu dahbo tlqun blon ahsbaxlah
92. kaxbaxdaxn tnszu dahbown lahlah
93. dahbown tnszu ahbaxbaxn faxlahw
94. gobaxzun ahbaxbaxn blon ahsbaxlah
95. tnsz dahbown gaxbaxre faxlah
96. tlq szahyun kaxbaxdaxn molah
97. tkus szahy gobaxzun gaxbaxrewn baxlahw

98. dahbown tlq whahwn faxlahw

99. ahbaxbax gaxbaxrewn lahlah

100. kaxbaxdax gaxbaxrewn tzaxlah

References

[1] X. Huang, A. Acero and H. Hon, *Spoken Language Processing*, 2001, Prentice Hall Inc., Upper Saddle River, New Jersey.

[2] L.R. Rabiner, and Biing-Hawand Jung, Fundamentals of Speech Recognition, 1993,pearson Education (singapor) pte.Ltd., Indian branch, India.

[3] *B.H. Juang & Lawrence R. Rabiner, Automatic Speech Recognition – A Brief History of the Technology Development, 2003, Georgia Institute of Technology, Atlanta*

[4] *Mark Hasegawa-Johnson, Lecture Notes in Speech Production, Speech Coding, and Speech Recognition, 2000, University of Illinois, Urbana-Champaign.*

*[5] Ronald A. Cole, Joseph Mariani, Hans Uszkoreit, Annie Zaenen, Victor Zue, Survey of the State of the Art in Human Language Technology, 1996, Sponsors***:***National Science Foundation European Commission*

[6] Daniel Jurafasky and James H.martin, Speech and Language Processing, 2000, Prentice-Hall,Inc., New Jersey.

[7] Roberto Gemello, Franco Mana, Dario Albesano Loquendo, Hybrid HMM/Neural Network based Speech Recognition in Loquendo ASR, pdf file

[8] Sisay Fissaha and Johann Haller , Application of corpus-based techniques to Amharic texts , Institute for Applied Information Sciences– University of Saarland Martin-Luther-Str.14, D-66111, Saarbrücken, Germany

[9] Solomon Teferra Abate and Wolfgang Menzel, Syllable-Based Speech Recognition for Amharic, Uniformity of Hamburg, Department of Informatik Natural Language Systems Groups, Vogt-Kölln-Strasse. 30, D-22527 Hamburg, Germany

[10] W.H. Abdulla, *Auditory Based Feature Vectors for Speech Recognition Systems*, 2002, Electrical and Electronic Engineering Department of Auckland University, Auckland, Newzealand.

[11] A. Alemu, L. Asker and M. Getachew, \Natural Language Processing for Amharic: Overview and Suggestions for a Way Forward," *Int. conf. on Automatic Treatment of Natural Languages*, 2003, Batz, France.

[12] P. T. Daniels and W. Bright, *The World's Written Languages*, 1996, Oxford University Press, Oxford, UK.

[13] S. Dong and M. Ku, *Finite State Network Decoder Based on Language Model Network*, 2001, Department of Computer Science, Sogang University, Seoul, Korea.

[14] B. Gold and N. Morgan *Speech and audio signal processing*, 2000, John Wiley & Sons, Inc.

[15] M. H. Hayes *Statistical Digital Signal Processing and Modeling*, 1996, John Wiley & Sons, Inc.

[16] B. Heine and D. Nurse, *African Languages*, 2000, Cambridge University Press, Cambridge, UK

.

[17] J. Mariani \Recent Advances in Speech Processing" Acoustics, Speech, and Signal Process ing, *Proc. of ICASSP-89*, International Conference on, 23-26 May 1989 , pp. 429-440 vol.1

[18] J. Schalkwyk, P. Hosom, Ed Kaiser and K. Shobaki, *CSLU-HMM: The CSLU Hidden Markov Modeling Environment*, 2000, Center for Spoken Language Understanding (CSLU), Oregon Graduate Institute of Science and Technology, Oregon, USA

[19] T. A. Solomon, W. Menzel, B. Ta¯la, *An Amharic Speech Corpus for Large Vocabulary Continuous Speech Recognition*, 2003, 15th International Conference on Ethiopian Studies, Hamburg, Germany.

[20] G. Strang, *Introduction to Applied Mathematics*, 1986, Wellesley-Cambridge Press, Wellesley, Massachusetts.

[21] M. Woszczyna, *Fast Speaker Independent Continuous Speech Recognition*, 1998, University of Karlsruhe: Institute of Logic, Complexity and Deductive Systems, Dissertation, Karlsruhe, Germany. 47 48 *BIBLIOGRAPHY*

[22] S. Young, et al., *The HTK Book*, 2002, Cambridge University Engineering Department, Cambridge, UK.

[23] *Automatic Speech Recognition*, Retrieved July 22, 2007 from the Mississippi State website: http://www.isip.msstate.edu/thesiss/speech/software.

[24] *Automatic Speech Recognition at CSLU*, Retrieved July 21, 2007 from the web site of CSLU: http://cslu.cse.ogi.edu/asr.

[25] *Ethiopian Census*, 1998, Central Statistics Authority of Ethiopia, Addis Ababa, Ethiopia.

[26] *Network Decoding*, Retrieved July 15, 2007 from the Mississippi State web site: http://www.isip.msstate.edu/thesiss/speech/software/tutorials/ monthly/2002/2002 04 network training/

[27] *What is HTK?*, Retrieved July 21, 2003 from the Hidden Markov Model Toolkit web site: http://htk.eng.cam.ac.uk/

[28] B. Yimam, *የአማርኛ ሰዋስው የተሻሻለ ሁለተኛ እትም*, 2000 E.C. እሌኒ ማ.ኃ.የተ.የግ.ማኅበር, Addis Ababa

[29] G. Amare, *ዘመናዊ የአማርኛ ሰዋስው በቀላል አቀራረብ*, 1989 E.C. ንግድ ማተሚያ ድርጅት, sixth edition, 2000 E.C. Addis Ababa

Index

Declaration

I, the undersigned student, declare that this thesis work is my original work, has not been presented for a degree in this or any other universities, and all sources of materials used for the thesis work have been fully acknowledged.

Name: Mesfin Birile Signature: _____

Place: Addis Ababa

Date of submission: July, 2008

This thesis has been submitted for examination with my approval as university advisors.

Molalgne Girmaw Signature: _____

ADVISOR

Manoj V.N.V Signature: _____

ADVISOR