ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES
FACULTY OF INFORMATICS
DEPARTMENT OF INFORMATION SCIENCE

APPLICATION OF AMHARIC SPEECH RECOGNITION SYSTEM TO COMMAND AND CONTROL COMPUTER: AN EXPERIMENT WITH MICROSOFT WORD

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

BY
MARTHA YIFIRU TACHBELIE
JULY, 2003

Acknowledgments

First of all, I should thank my God for supporting and being with me in all walks of my life. God, I thank you so much.

My sincere gratitude should go to my advisors Ato Tesfaye Birru, Ato Solomon Berehanu, Ato Kinfe Taddesse and Dr. Moges for their constructive comments and guidance. I am grateful to them because without their guidance and comments the completion of this research would have not been possible. I am also indebted to Ato Sebisibe H/Mariam for providing constructive comments.

I would like to thank those people who spent their precious time to provide their voice for use in the experiment. I really thank them.

My special thanks should go to Ato Solomon Teferra for his love, encouragement, comment and kindness to help me in any way. I should also thank him for typing this thesis.

All the SISA staff members deserve my sincere thanks, for their encouragement and support throughout my study.

Last, but most important, thanks go to my family; specially to my mother W/ro Asrat Demeke for her love and encouragement without which I would have not been who I am today.

Abstract

This study explored the possibility of developing Amharic speech input interface to command and control Microsoft Word. Towards this end, literature were reviewed on speech recognition, application of speech recognition, HMM and its application in speech recognition, Amharic speech recognition, HTK and human-computer interaction.

Speech input interface requires speech recognition system. To develop and test the required Amharic speech recognition system speech data were recorded from 26 people (10 female and 16 male) in the age range of 20 to 35. 76.9% of the recorded data were used to train the recognizers and the remaining data were used for testing the performance of recognizers. Two (fixed variance and variable variance based models) HMM-based, speaker independent, small vocabulary, isolated Amharic word recognizers were developed. The performance of these recognizers was tested using the test data. Although both of them recognized all the test data correctly, the performance of recognizer with variable variance performed better than the recognizer with fixed variance in live environment. Thus, the recognizer with variable variance was further considered for the development of the prototype Amharic speech input interface system.

Speech input interface requires communication interface that sends the recognized command word to the application, Microsoft Word in this case. In this study the communication interface was written using Visual Basic 6. To test the performance of the system as a whole, 18 randomly selected command words were given to 6 people (3 command words for each) and these people were asked to command Microsoft Word orally. The system performed 16 commands accurately and only two command words were wrongly recognized and thus Microsoft Word performed wrong actions. Finally, the prototype speech input interface system developed in this experiment was integrated with Microsoft Word as a Macro for regular use.

The result of the experiment showed the feasibility of developing Amharic speech input interface. Based on the result of the study and what has been learned in the course of the research recommendations for further study were forwarded.

Table of Contents

List of figures

CHAPTER ONE

INTRODUCTION

1.1. Background

Human beings have been dreaming of and struggling for developing intelligent machines. An important function of these machines is that they master speech input and output to communicate with us naturally. They understand human speech and generate human like speech messages (Chollet, 1994). Developing such machines is important since, as indicated by Nahm and Slater (1997), speech interface in a user's own language is an ideal means of communication for being the most natural, flexible, efficient and convenient option that frees hands and eyes for other tasks.

Due to the advancements made in digital signal processing, pattern matching, classification algorithms and computer hardware technology, the dream of providing speech interface to computers has become a reality (Karl, Pettey and Shneiderman, 1993). Supporting this idea, Adams et. al.(1999) indicated that speech technology has advanced to "the stage where it offers great promise for human-computer interaction in a variety of applications."

Earlier, people considered speech as a universal medium and better in all aspects than all other media for human-computer interaction (Ibid). However, various researches disproved this idea. Lefebvre, Duncan and Poirier (1993) said that there are various forms of communication that can be utilized in human-computer interaction but no medium is sufficient by itself. Karl , Pettey and Shneiderman (1993) also said that "speech input is not likely to replace other modes of input, but seems to have a useful place along with other

types of input in a multimodal interface." Thus the attention of research community is changed toward the employment of speech input and output in multimodal interface (Adams et. al., 1999).

Multimodal interaction refers to the use of more than one medium for human-computer communication. Different modalities have complementary advantages and disadvantages and can be used together to develop multimodal interfaces that can compensate weaknesses of one interface via strengths of another (Cohen and Oviatt, 1994). Cohen and Oviatt argue that multimodal interface provide the following advantage.

- Enhanced error avoidance and correction – multimodal interface offer the opportunity for users to avoid errors that would otherwise occur in a unimodal interface
- Accommodation of various situation, users and task. The environment may change people's preference to employ one modality of communication over the other. In addition, individual and task differences can influence people's preference to use one mode over another
- User preference. Users strongly prefer multimodal interaction.

Speech interface, especially if it is speech only, requires both speech recognition and speech synthesis systems. Speech recognition system is used as an input system whereas speech synthesis is used in speech output. However, in multimodal interface, one of them may not be required. In a multimodal interface which uses only speech input we need only

speech recognizer while an interface that provides output through speech needs only speech synthesizer. Furui (1995) indicated that "input by speech recognition and output by text and graphics is an ideal combination in most interactive systems..."

As indicated by Sun Microsystems (1998), speech input can be used when:

- No keyboard is available e.g. over the telephone, portable devices.
- Task requires the user's hands to be occupied so they can not use other modalities such as keyboard and mouse.
- Commands are embedded in a deep menu structure.
- Users are unable to type or are not comfortable with typing.
- Users have a physical disability.

Adams et. al. (1999) also stated that direct speech control is essential when interacting with a system in the dark, while one's hands are occupied or if one is handicapped or at a remote site. He also stated that "even where keyboard and mouse would be accessible, their limited fit to human capabilities means that it may be more natural and comfortable to interact conversationally by speech."

Research has not proven the effectiveness of speech recognition as a general purpose input mode. But, speech input can be useful in certain situations (Christian et. al., 2000). A study by Karl, Pettey and Shneiderman (1993) indicated that using speech to issue commands to word processing applications, while using the mouse for direct manipulation and keyboard for text entry, speed up task time. Commenting on the result of Karl, Pettey and

Shneiderman's (1993) study, Christian et. al. (2000) said that in Karl, Pettey and Shneiderman's (1993) study the number of commands that could be issued through speech is relatively small and the user spoke very short sentences. Van Buskirk and Lalomia (1995) said that "the best tasks for speech input were tasks in which the user has to issue brief commands using a small vocabulary."

Speech Recognition (SR), as described by Junqua and Haton (1996), is the "decoding of the information conveyed by a speech signal and its transcription into a set of characters." The resulting characters can be used to perform various tasks such as controlling a machine, accessing a database, or producing a written form of the input speech. In this study, the possibility of using the resulting characters of speech recognition system to command and control Microsoft Word was explored.

Types of Automatic Speech Recognition Systems

Depending on the flow of speech, size of vocabulary and number of speakers, speech recognizers can be categorized into different groups. These groups are:

- Discrete versus continuous speech recognition systems.
- Large versus small vocabulary systems.
- Speaker-dependent versus speaker independent systems.

In discrete or isolated word recognizers, a user must pause between words - that means artificial pause should be inserted before and after each word. As indicated by Lea (1982) this helps the system to identify word boundaries easily. In addition, co-articulation effect

parametric representation of speech are highly speaker dependent, and a set of reference patterns suitable for one speaker may perform poorly for another speaker. Speaker dependent systems, on the other hand, recognize speech from those people whose speech is used during the development of the recognizer, thus avoiding most of the speaker variability problem (Junqua and Haton, 1996).

1.2. Statement of the Problem and its Justification

The ultimate aim of research in speech technology is the development of human-computer conversational system that communicates with any one, about any thing, on any topic and in any situation (Markowitz, 1996). To achieve this goal, researches have been conducted for about 50 years and various systems are developed for various languages. For example, Microsoft Corporation developed English speech interface that enables people to command and control applications in Microsoft Office and dictate what they want to write instead of using keyboard.

However, there are few researches on speech technology in Ethiopian languages in general and Amharic in particular. Laine (1998) and Morka (2001) made efforts to develop an Amharic and Oromiffaa text-to-speech synthesis systems, respectively. Solomon (2001) and Kinfe (2002) developed Amharic consonant vowel (CV) syllable and subword-based isolated word recognizers, respectively. Both workers developed speaker dependent and speaker independent systems using HMM.

Amharic is a dominant language in Ethiopia, the official language of the Federal Government of Ethiopia and the most commonly learned second language through out the country (Bender et. al., 1976). However, as to the knowledge of the researcher, there is no research conducted on the application of Amharic speech technology in general and Amharic speech input interface in particular.

Microsoft Word is word processing software that is most widely used to create and process documents in Ethiopia. However, it does not provide Amharic speech input interface as it does for English. On the other hand, it is a human behavior to prefer speaking in their mother tongue, which is true for Amharic speakers. Consequently, Amharic speaking people could not get the benefit that could be obtained from a speech input interface system. Thus, the need for a system that enables Amharic speakers to command and control Microsoft Word through speech interface is unquestionable.

To this end, the purpose of this study was to explore the possibilities of developing Amharic speech input interface that helps Amharic speakers to command and control Microsoft Word.

Developing speech input interface system requires speech recognition system. The appropriate speech recognition system for command and control application is a speaker-independent, small vocabulary, isolated word system.

Markowitz (1996) indicated the appropriateness of speaker independent recognition system (which can be achieved by the use of large number of training speakers (Young et. al., 2002)) for applications – such as speech input interface. The appropriateness of isolated speech recognition system for command and control applications (in which the user is required to speak command words one at a time) is indicated by Rabiner and Juang (1993). Moreover, although the vocabulary size varies according to the operation performed by the equipment or the software for which the command and control system is designed, most command and control applications require a small vocabulary recognition system (Markowitz, 1996). Although whole word modeling is not practicable for large vocabulary systems, in small vocabulary system it is possible to model words instead of subword units that lead to degraded performance (Lee, 1989).

The results of the two researches conducted on Amharic speech recognition (Solomon, 2001 and Kinfe, 2002) could not be used in this study directly. The work of Solomon could not be used in this research since it recognizes only Amharic CV syllables while word recognizer is required in this study. Kinfe (2002), developed speaker-independent and speaker-dependent subword based Amharic isolated word recognizers. He used subword units with the aim of having large vocabulary system. However, he did not include all phones and all CV syllables of Amharic language in his experiment and, therefore, adding new words that are required to command and control Microsoft Word to his dictionary is difficult. Moreover, the performance of his recognizers is not high enough for command and control purpose. Therefore, HMM based speaker-independent, small vocabulary, isolated word speech recognition system was developed. Since the system is small

vocabulary, one HMM was developed for each command word. In other words, whole-word modeling was employed.

1.3. Benefits of the study

Amharic speech input interface is very helpful for handicapped Amharic speakers that means for users who have difficulty in using their hands to type, but are able to speak clearly. In addition, blind users can use this kind of system since they have difficulty in using keyboard and mouse to command and control computers. Other group of users that can get benefit from this kind of system is people whose eyes and hands are busy in performing other task. In general, it can be said that such system is helpful for any people who can speak Amharic. This study is, therefore, a step towards the development of such a useful system.

The result of this study can also be used as an input towards the development of human-computer conversational system.

1.4. Objective of the Study

The general and specific objectives of the study are the following.

1.4.1. General Objective

The general objective of the study is to explore the possibility of developing Amharic speech input interface to command and control Microsoft Word.

1.4.2. Specific Objectives

The specific objectives of the study are:

- Identify and select English command words used in Microsoft Word and translate them to Amharic.
- Build a prototype speaker-independent small vocabulary Amharic isolated word recognition system.
- Test the performance of the prototype recognizer.
- Develop an interface that communicates the recognized Amharic commands to Microsoft Word.
- Test the usability of the speech interface.
- Forward recommendation for further study.

1.5. Methods

The following methods were employed in conducting the study.

1.5.1. Data Collection techniques

Literature Review

Extensive literature review was conducted on speech recognition, application of speech recognition, human-computer interaction, Hidden Markov Model and its application in speech recognition. Moreover, researches conducted on Amharic speech recognition are reviewed. To understand how the HTK (Hidden Markov Model Toolkit) works the HTK book and various other sources were consulted.

Data Preparation

Considering the available time and financial resource for this research, only fifty command words used to command and control Microsoft Word were selected, translated to Amharic and used to develop the prototype system. Speech data of these words was recorded from 26 people with the age range of 20 to 35. The speech data was divided into two parts: training (20 speakers) and test set (6 speakers). The training set was used to train the recognizer while the test set was used to test the performance of the recognizer.

1.5.2. Modeling Technique

Prototyping approach has been used in the course of developing the speech input interface. Prototype Amharic isolated word recognizer was built using Hidden Markov Model that became the predominant technique for speech recognition (Lee, 1989). For this purpose, HTK (Hidden Markov Model toolkit) has been employed. This toolkit was preferred since both Solomon (2001) and Kinfe (2002) had used the toolkit and achieved considerable results. In addition the toolkit is freely available for academic and research use.

To develop the required communication interface, Visual Basic 6.0 programming language has been used. This programming language is preferred over others because it has good string manipulation feature and it is user friendly.

1.5.3. Testing Technique

The recognition system must be evaluated or tested for its accuracy. To test the recognition system, the HTK toolkit was used and then the accuracy was reported using table and percentages as required.

To test the performance of the recognizers in live recognition, 5 (3 of them were involved in training the recognizers while 2 of them were not) individuals were chosen based on availability and asked to utter all command words used in the study.

After developing the communication interface that works between the recognizer and the application, the system was evaluated by allowing 3 male and 3 female users to use the speech input interface. Each user was asked to pass three commands to Microsoft Word orally and the performance of the system was reported.

1.6. Scope and Limitation of the Study

The aim of the study was to explore the possibility of developing Amharic speech input interface system to command and control Microsoft Word. However, due to time and financial constraints only 50 command words are considered in the study. Moreover, only the first command could be passed to tasks that require multiple commands. For example, the command open (ክፈት) brings only the open dialog box after which the user should use mouse or keyboard to select a file, open the file or cancel the operation.

This research can not be claimed to be complete and it has the following limitations.

- One problem of speech input interface is recognition error. This system has no error handling mechanism.
- Since it is difficult to record more data with the available time, training data was collected only from 20 readers although training model with large data increases a speaker independent feature of the recognizer.

1.7. Organization of the Thesis

This paper is divided into 5 chapters. Chapter one consists of background, statement of the problem and its justification, objectives of the study, methodology followed in the course of the study and the scope and limitations of the study. In chapter two speech recognition is reviewed. Chapter three presents human computer interaction and application of speech recognition. Chapter four provides detail description of the prototype speech input interface system. Finally, conclusions and recommendations are given in chapter five.

Chapter Two

Basics of Speech Recognition

Emphasizing speech recognition system that plays the role of human-computer interface, Markowitz (1996) indicated that such system performs three primary tasks: preprocessing, recognition and communication. This chapter provides a brief description of these tasks.

2.1. Preprocessing

Speech is an analog signal - "a continuous flow of sound waves and silence." This form cannot be processed directly by a speech recognizer. Thus, there is a need for preprocessing the speech signal. Preprocessing is the conversion of spoken input into the form that is acceptable by the recognizer. Specifically, it deals with digitizing analog speech and converting the speech wave form to some type of parametric representation (Markowitz, 1996). Analog to digital conversion is done using analog to digital converters (Markowitz, 1996) whereas conversion to parametric representation is done by digital signal processing front ends (Rabiner and Juang, 1993). Rabiner and Juang (1993) also stated that, although there are many possibilities for parametrically representing the speech signal, short time spectral envelope is the most important. Therefore, short time spectral analysis methods are considered as the core of the signal processing front end.

The two most commonly used and dominant spectral analysis methods are filter bank and linear predictive coding (LPC) (Rabiner and Juang, 1993; Markowitz, 1996). Rabiner and Juang also said that LPC and filter bank have proven themselves to give the highest performance in most practical speech recognition systems.

LPC is predicated on the idea that a speech sample can be approximated as a linear combination of past speech samples. Rabiner and Juang (1993) stated that LPC is one of the most powerful and dominant speech coding techniques. It provides a reliable, accurate and robust method of estimating parameters that characterize speech signal. The importance of LPC lies in its ability to provide extremely accurate estimates of speech parameters, in its speed of computation and less storage requirement. Markowitz (1996) also indicated the fact that these advantages of LPC helped it to serve as the basis for other forms of coding such as cepstral coefficients and vector quantization.

On the other hand, as indicated by Young et. al. (2002), the human ear resolves frequencies non-linearly across audio spectrum. Moreover, as Young et. al. said, empirical evidences suggest that designing a front end that operates in a non-linear manner improves recognition performance. However, LPC (as its name implies) does not consider non-linearity. This makes filter bank a popular alternative to LPC because it provides a way of obtaining the desired non-linearity. However, to use the resulting data in Hidden Markov Model based recognizers, use of cepstral transformation is a must since filter bank amplitudes are highly correlated. It is important to note that HTK supports both spectral analysis methods. But, this study uses filter bank in the course of developing speech recognizer because of its capability of resolving frequencies non-linearly.

2.2. Recognition

The step that follows preprocessing is recognition at which the system identifies what has been said. As indicated by Rabiner and Juang (1993), there are three approaches to automatic speech recognition. These are: acoustic-phonetic, pattern recognition and artificial intelligence. A description of these approaches is given below.

2.2.1. Acoustic-Phonetic Approach

Rabiner and Juang (1993) stated that acoustic-phonetic approach is based on the theory of acoustic phonetics. They said that this theory "postulates that there exist finite, distinctive phonetic units in spoken language and that the phonetic units are broadly characterized by a set of properties that are manifest in the speech signal, or its spectrum, over time." It has two steps (segmentation and labeling, and word level recognition) besides the preprocessing step which is common to all other approaches (Markowitz, 1996; Rabiner and Juang, 1993).

Segmentation and labeling involves segmenting the speech signal into discrete regions and attaching one or more phonetic labels to each segmented region on the basis of the observed acoustic properties. In other words, this is a step where the system tries to get regions where the features change very little (stable regions) and label the segmented region. This step is the heart of acoustic-phonetic approach and it is the most difficult task to carry out reliably. Because of this difficulty, various control strategies are used to limit the range of segmentation and label possibilities. For example, one may impose a constraint that says a word should contain at least two and at most six phonetic units. Another

constraint may be to consider only words with n phonetic units when the segmentation provides n – 1 segmentation points. These kinds of control strategies will reduce the search space and increase the performance of the system (Rabiner and Juang, 1993).

In segmentation and labeling step, there is uncertainty resulting from high degree of acoustic similarity among phonemes and phoneme variability – caused by co-articulation effects and other sources (Markowitz, 1996). Due to this uncertainty, the result of segmentation and labeling step is a set of phoneme hypothesis usually organized into a phoneme lattice. This phoneme lattice is an input to the second step of acoustic-phonetic approach. The second step – word level recognition – attempts to determine a valid word or string of words from the sequence of phonetic labels (Rabiner and Juang, 1993; Markowitz, 1996).

Rabiner and Juang (1993) said that although acoustic-phonetic approach is an interesting idea, it lacks success in practical speech recognition systems due to many problems associated with it. They mentioned the following problems.

- The difficulty of decoding phonetic units into word string (Lexical access problem).
- The difficulty of getting reliable phoneme lattice for the lexical access stage.
- The requirement of extensive knowledge of the acoustic properties of phonetic units.
- The choice of features is made mostly based on ad-hoc considerations.
- The design of sound classifiers is also not optimal.
- No well defined, automatic procedure exists for tuning the method e.g. adjusting

decision thresholds etc., on real, labeled speech. There is no ideal way of labeling the training speech in a manner consistent and agreed on uniformly by linguistic experts.

2.2.2. Pattern Recognition Approach

Pattern recognition approach to speech recognition, unlike acoustic-phonetic approach, is the one in which the speech patterns are used directly without explicit feature determination and segmentation. This method involves two steps: training of speech patterns and recognition of the pattern via pattern comparison. During training phase speech knowledge is brought into the system. The idea behind the training phase is that if enough versions of a pattern to be recognized are included in the training set that is provided to the algorithm, the training algorithm should be able to adequately characterize the acoustic properties of the pattern. In the second step – pattern recognition – unknown speech or the speech to be recognized is compared with each possible pattern learned in the training phase and classified according to the goodness of match of the patterns (Rabiner and Juang, 1993).

Pattern recognition approach is the method of choice for speech recognition because of the following reasons (Ibid).

1. Simplicity of use. Pattern recognition approach is easy to understand and it is the most widely used method. Moreover, it is rich in mathematical and communication theory justification for procedures used in training and decoding.
2. Robustness and invariance to different speech vocabularies, users, feature sets, pattern comparison algorithms and decision rules. Because of this, pattern recognition approach is appropriate for different kinds of speech units (e.g.

Phoneme, syllable, word, phrase, sentence, etc.), word vocabulary, speaker populations, background environment, transmission condition, etc.

3. Proven high performance. This approach provides high performance on any task that is reasonable for the technology.

Pattern-recognition approach can further be classified into different types such as template matching and stochastic depending on such factors as the type of feature measurement, the choice of template or models for reference patterns, and the method used to create reference patterns and classify unknown test pattern (Rabiner and Juang, 1993). In stochastic processing, for instance, there is no direct matching between stored model and input, unlike template matching approach. Instead, it is based upon complex statistical and probabilistic analyses, which are best understood by examining the network like structure in which those statistics are stored (Markowitz, 1996). Markowitz (1996) noted that stochastic approach is the most commonly used approach to speech recognition.

The term stochastic refers to "the process of making a sequence of non-deterministic selections from among sets of alternatives" (Markowitz, 1996). It consists of employing a probabilistic model for the uncertainty or incomplete information that is inherent in speech signals (Baker, 1990). Stochastic modeling requires the creation and storage of models for each item that will be recognized. However, stochastic modeling involves no direct matching between stored models and the input. Instead, as indicated above, it is based on complex statistical and probabilistic analyses. A well known and most widely used stochastic model is the Hidden Markov Model.

2.2.2.1. The Hidden Markov Model (HMM)

The theory of Hidden Markov Model and its application to speech recognition is not new. The basic theory was published in a series of papers by Baum and his colleagues in the late 1960s and early 1970s and was implemented for automatic speech recognition independently at CMU and IBM in the 1970s (Rabiner, 1990; Lee, 1989).

Hidden Markov Model (HMM), which is also called Markov sources or probabilistic functions of Markov Chains, is one type of stochastic model (Rabiner, 1990; Rabiner and Juang, 1993). HMM is defined by Young et. al. (2002) as "a finite state machine which changes state once every time unit and each time t that a state j is entered, a speech vector o_t is generated from the probability density $b_j(o_t)$." They also indicated the fact that transition from state i to state j is probabilistic and is governed by discrete probability a_{ij}. In HMM we only observe the output sequence whereas the state sequence is hidden. That is why the model is called hidden Markov model (Lee, 1989).

As indicated by Rabiner and Juang (1993) HMM is defined by the following elements.

- N, the number of states in the model. Individual states are labeled as {1, 2,…, N} and the state at time t is denoted by q_t.
- M, the number of distinct observation symbols per state. Observation symbols correspond to the physical output of the system being modeled. Individual observation symbols are denoted as $V = \{v_1, v_2, \ldots, v_m\}$
- The state transition probability distribution $A=\{a_{ij}\}$ where

$$a_{ij} = p(q_{t+1} = j / q_t = i), \; 1 \le i, \; j \le N$$

- The observation symbol probability distribution, B=$\{b_j(k)\}$ in which

$$b_j(k) = p(o_t = v_k / q_t = j),\ 1 \le k \le M$$

$b_j(k)$ defines the symbol distribution in state i, j=1,2,…,N

- The initial state distribution $\pi = \{\pi_i\}$ in which

$$\pi_i = p(q_1 = i),\ 1 \le i \le N$$

Since a_{ij} and $b_j(k)$ are probabilities they satisfy the following probability rules as indicated by Lee (1989):

$$a_{ij} \ge 0,\ \forall i, j$$

$$b_j(k) \ge 0,\ \forall j, k$$

$$\sum_j a_{ij} = 1,\ \forall i$$

$$\sum_k b_j(k) = 1,\ \forall j$$

There are different types of HMM. As indicated by Rabiner and Juang (1993), on the basis of the structure of the transition matrix, HMM can be classified as ergodic/fully connected and left-to-right/Bakis. Ergodic is an HMM in which "every state of the model can be reached from every other state in a finite but aperiodic number of steps". This kind of model has a property that all a_{ij} coefficients are positive. In left-to-right model, as time increases the state index increases (or stays the same). In addition to this, backward transition is not possible, i.e. a_{ij} = 0, when j<i. To make sure that large changes in state

indices do not occur, additional constraint is placed on a_{ij}, which is given in the following form:

$$a_{ij} = 0, j > i + \Delta i,$$

where Δi is the maximum number of states to be jumped. If the value of Δi is 2 for example, it means that no jumps of more than two states are allowed. Left-to-right HMM is appropriate for modeling speech signal since its properties change over time in a successive manner.

HMMs can also be classified as discrete and continuous density (Lee, 1989). Discrete density HMM is an HMM in which the observations are considered as discrete and consequently discrete probability density is used. Since speech observations are continuous in nature, various methods have been used to discretize the speech signal. One of these methods is vector quantization. However, discretization introduces serious signal degradation (Lee, 1989; Junqua and Haton, 1996; Rabiner and Juang, 1993). Hence, as indicated by Rabiner and Juang it would be advantageous to be able to use continuous density HMM – which is HMM with continuous observation densities. This type of HMM models continuous signal directly. HTK provides tools for building both types of HMMs.

When HMM is used in speech recognition, it can model any kind of speech unit. It can model subword units (like phoneme, syllable, etc.), a word or a sentence. In large vocabulary recognition systems, to limit the amount of training data and storage required for modeling words, HMM usually represents subword units. Conversely, in small vocabulary systems, HMM is mostly used to model words (Junqua and Haton, 1996).

HMM for isolated word recognition

The idea presented in this section is taken from Young et. al. (2002) unless specific names are given.

For isolated word recognition, HMM is trained for each word using different examples. Then to recognize unknown word, the probability of each model generating that word is computed and the most probable model identifies the word. This idea is depicted by the following figure:

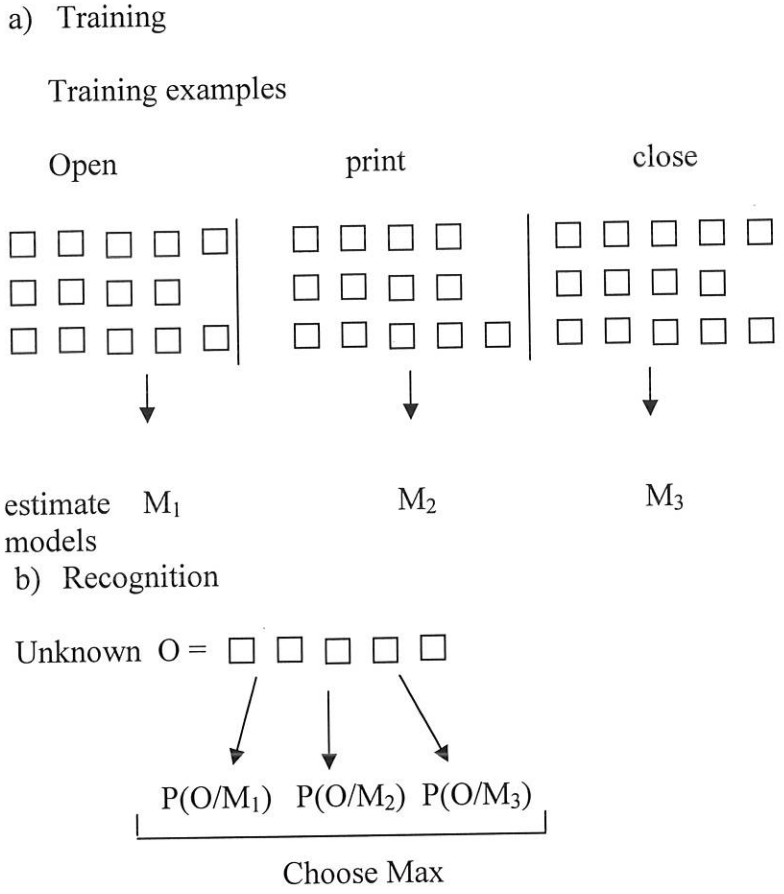

fig.2.1. Using HMM for isolated word recognition (adapted from Young et.al. (2002)).

The three problems of HMM and Their Solutions

To use HMM practically, there are three basic problems that must be solved (Lee, 1989; Rabiner and Juang, 1993; Junqua and Haton, 1996). These are:

- The evaluation problem: given a model and sequence of observation on the speech signal, what is the probability of observation sequence given the model?
- The decoding problem: given the model and a sequence of observation, what is the state sequence that best explains the observation?
- The learning problem – given an HMM and a sequence of observation, how to adjust the model parameters to maximize the probability of generating the observation?

The following is a description of the solutions of the above three problems.

The Forward Algorithm

The evaluation problem, as indicated by Lee (1989), is stated as: given a model M, compute the probability that the model will generate a sequence of observation, O = $(o_1,o_2,\ldots,o_T)$, i.e. computing p(O/M). Rabiner and Juang (1993) said that the "most straightforward way of doing this is through enumerating every possible state sequence of length T (the number of observations)."

Let q be state sequence given as $(q_1q_2q_3\ldots q_T)$, where q_1 is the initial state. As indicated by Rabiner and Juang (1993), the probability of O given q is defined as

$$p(O/q,M) = \prod_{t=1}^{T} p(o_t / q_t, M) \qquad 2.1$$

which can also be defined as follows by assuming statistical independence of observations.

$$p(O/q,M) = b_{q_1}(o_1)\, b_{q_2}(o_2) \cdots b_{q_T}(o_T) \quad 2.2$$

The probability of a state sequence, q can be written as (Ibid):

$$p(q/M) = \pi_{q_1} a_{q_1 q_2}\, a_{q_2 q_3} \cdots a_{q_{T-1} q_T} \quad 2.3$$

Although the state sequence, q is hidden and only the observation sequence O is known (Young et. el., 2002), the joint probability that O is generated by the model M as it moves through q can be given as a product of the above two terms (Rabiner and juang, 1993) i.e.

$$p(O,q/M) = p(O/q,M).p(q/M) \quad 2.4$$

Since q is hidden, the required probability, p(O/M), is calculated by summing over all possible state sequence, q (Young et. el., 2002). Thus,

$$p(O/M) = \sum_{q} p(O/q,M)p(q/M) \quad 2.5$$

$$p(O/M) = \sum_{q_1 q_2 \cdots q_T} \pi_{q_1} b_{q_1}(o_1) a_{q_1 q_2} b_{q_2}(o_2) \cdots a_{q_{T-1} q_T} b_{q_T}(o_T) \quad 2.6$$

Calculation of p(O/M) using the above equation involves order of $2T.N^T$ computation, where T is the number of observations at each state and N is the number of states. Thus, calculating p(O/M) using the above equation is computationally not feasible (Rabiner and Juang, 1993). However, p(O/M) can be computed efficiently using simple recursive procedure – the forward algorithm (Young et. el., 2002).

Rabiner and Juang (1993) described the forward algorithm as follows.

Let $\alpha_t(i)$ be the forward variable defined as:

$$\alpha_t(i) = p(o_1 o_2 \cdots o_t, q_t = i/M) \quad 2.7$$

which is the probability of the partial observation sequence $o_1 o_2 \ldots o_t$ and state i at time t, given the model. The value for $\alpha_t(i)$ is solved as follows.

1. Initialization

$$\alpha_1(i) = \pi_i b_i(o_1), \quad 1 \le i \le N \qquad 2.8$$

2. Induction

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

$$1 \le j \le N$$

3. Termination

$$p(O/M) = \sum_{i=1}^{N} \alpha_T(i) \qquad 2.10$$

Using the forward algorithm the computation involved in the calculation of $\alpha_t(i)$, where $1 \le t \le T$ and $1 \le j \le N$, is on order of N^2T rather than $2TN^T$. That means the forward algorithm enables us to evaluate p(O/M) efficiently. However, according to Lee (1989), in speech recognition the interest is in finding p(M/O). To get this we can use Bayes rule, which is defined as:

$$p(M/O) = \frac{p(O/M)p(M)}{p(O)} \qquad 2.11$$

The Viterbi Algorithm

Decoding problem deals with, given a model and an observation sequence, finding the most likely or optimal state sequence in the model that produced the observation sequence. However, state sequence is hidden in an HMM. Thus, to solve the problem it is possible to produce the "state sequence that has the highest probability of being taken while generating the observation

sequence." To do this we use Viterbi algorithm, which is a modification of forward algorithm. Instead of summing probabilities that came together as in the forward algorithm, in Viterbi we need to choose and remember the maximum probability (Lee, 1989).

The following idea on Viterbi algorithm is taken from Rabiner and Juang (1993).

To find best state sequence $q(q_1q_2\ldots q_T)$, given observation sequence $O = (o_1o_2\ldots o_T)$, we need to define the quantity

$$\delta_{t+1}(i) = \max_{q_1 q_2 \cdots q_{t-1}} p(q_1 q_2 \cdots q_{t-1}, q_t = i, o_1 o_2 \cdots o_t / M) \qquad 2.12$$

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

$$\delta_{t+1}(j) = \left[\max_i \delta_t(i) a_{ij}\right] b_j(o_{t+1}) \qquad 2.13$$

Generally, the complete procedure of Viterbi algorithm is given as follows.

Let $\psi_t(j)$ be an array used to keep track the argument that maximized equation 2.13 for each t and j. $\psi_t(j)$ is used to retrieve the best state sequence.

1. Initialization

$$\delta_1(i) = \pi_i b_i(o_1), \quad 1 \le i \le N \qquad 2.14$$

$$\psi_1(i) = 0 \qquad 2.15$$

2. Recursion

$$\delta_t(j) = \max_{1 \le i \le N} \left[\delta_{t-1}(i) a_{ij}\right] b_j(o_t), \quad 2 \le t \le T \ , 1 \le j \le N \qquad 2.16$$

$$\psi_t(j) = \arg\max_{1 \le i \le N} \left[\delta_{t-1}(i) a_{ij}\right], \quad 2 \le t \le T \ , 1 \le j \le N \qquad 2.17$$

3. Termination

$$p^* = \max_{1 \le i \le N} [\delta_T(i)] \qquad 2.18$$

$$q_T^* = \arg\max_{1 \le i \le N} [\delta_T(i)] \qquad 2.19$$

4. State sequence backtracking

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \qquad t = T=1, T-2, \ldots, 1 \qquad 2.20$$

The Forward – Backward Algorithm

The third problem of HMM is the learning problem in which, given the model and an observation sequence, we attempt to adjust the model parameters to maximize the probability of generating the observation sequence. This problem is the most difficult problem since there is no known analytical method to solve for the model parameters that maximizes the probability of the observation sequence (Lee, 1989; Rabiner and Juang, 1993). Instead, an iterative procedure or gradient decent technique is used to solve the problem. One iterative procedure that is used to solve this problem is the forward – backward algorithm, which is also called Baum Welch algorithm (Lee, 1989). This algorithm is described as follows.

In forward procedure, the forward variable $\alpha_t(i)$ is defined. It is also possible to consider the backward variable, $\beta_t(i)$, which is the probability of the partial observation sequence from t+1 to the end, given state i at time t and the model M. $\beta_t(i)$ is defined as (Rabiner and Juang, 1993):

$$\beta_t(i) = p(o_{t+1} o_{t+2} \cdots o_T / q_t = i, M) \qquad 2.21$$

Like $\alpha_t(i)$, as indicated by Rabiner and Juang, $\beta_t(i)$ can be solved inductively as follows.

1. Initialization

$$\beta_T(i) = 1, \qquad 1 \le i \le N \qquad 2.22$$

2. Induction

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j), \qquad 1 \le i \le N \qquad 2.23$$

$$t=T-1, T-2, \ldots, 1$$

To describe the procedure for re-estimation of HMM parameters, $\varepsilon_t(i, j)$ which is the probability of being in state i at time t and state j at time t+1 should be defined first (Ibid). $\varepsilon_t(i, j)$ is defined as:

$$\varepsilon_t(i, j) = p(q_t = i, q_{t+1} = j / O, M) \qquad 2.24$$

$$\varepsilon_t(i, j) = \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{p(O / M)} \qquad 2.25$$

The probability of being in state i at time t, given the entire observation sequence and the model can be obtained by summing $\varepsilon_t(i, j)$ over j.

$$\gamma_t(i) = \sum_{j=1}^{N} \varepsilon_t(i, j) \qquad 2.26$$

Rabiner and Juang (1993) also said that summing $\gamma_t(i)$ over the time index t, gives the expected number of transitions from state i to any state at any time given O. Similarly,

summing $\varepsilon_t(i,j)$ over the time index t, gives the expected number of transitions from state i to state j. That is,

$\sum_{t=1}^{T-1}\gamma_t(i)$ = expected number of transitions from state I to any state at any time given O. 2.27

$\sum_{t=1}^{T-1}\varepsilon_t(i,j)$ = expected number of transitions from state I to state j given O. 2.28

It is, therefore, possible to compute model parameters (π, a_{ij} and $b_j(k)$) using equation 2.27 and 2.28 as follows (Ibid).

$$\overline{\pi}_j = \gamma_t(i) \qquad 2.29$$

$$\overline{a}_{ij} = \frac{\sum_{t=1}^{T-1}\varepsilon_t(i,j)}{\sum_{t=1}^{T-1}\gamma_t(i)} \qquad 2.30$$

$$\overline{b}_j(k) = \frac{\sum_{t=1}^{T}\ \gamma_t(j)}{\sum_{t=1}^{T}\gamma_t(j)} \qquad 2.31$$

Generally Lee (1989) put the forward-backward algorithm as follows.

1. Given an initial set of parameters $\{\pi, a_{ij}, b_j(k)\}$
2. Compute $\overline{\pi}$, $\overline{a}_{ij}$ and $\overline{b}_j(k)$ according to the re-estimation formulae in equations 2.29, 2.30 and 2.31.

summing $\varepsilon_t(i,j)$ over the time index t, gives the expected number of transitions from state i to state j. That is,

$\sum_{t=1}^{T-1}\gamma_t(i)$ = expected number of transitions from state I to any state at any time given O. 2.27

$\sum_{t=1}^{T-1}\varepsilon_t(i,j)$ = expected number of transitions from state I to state j given O. 2.28

It is, therefore, possible to compute model parameters (π, a_{ij} and $b_j(k)$) using equation 2.27 and 2.28 as follows (Ibid).

$$\overline{\pi}_j = \gamma_t(i) \qquad 2.29$$

$$\overline{a}_{ij} = \frac{\sum_{t=1}^{T-1}\varepsilon_t(i,j)}{\sum_{t=1}^{T-1}\gamma_t(i)} \qquad 2.30$$

$$\overline{b}_j(k) = \frac{\sum_{t:o_t=v_k}^{T}\gamma_t(j)}{\sum_{t=1}^{T}\gamma_t(j)} \qquad 2.31$$

Generally Lee (1989) put the forward-backward algorithm as follows.

1. Given an initial set of parameters $\{\pi, a_{ij}, b_j(k)\}$
2. Compute $\overline{\pi}$, $\overline{a}_{ij}$ and $\overline{b}_j(k)$ according to the re-estimation formulae in equations 2.29, 2.30 and 2.31.

2.2.4. Unit of Recognition

At the training phase of a speech recognizer, reference models are constructed. These models can be based on different units of speech (Rodman, 1999) such as phones, word-dependent phones, context-dependent phones, multi-phone and Words (Lee, 1989). Phrases can also be considered as a possible unit of recognition (Rodman, 1999). The unit of speech used in speech recognition should be trainable, well defined and relatively insensitive to context (Lee, 1989).

Phones, word-dependent phones, context dependent phones and multi-phone units are called subword units. Phone is trainable since there are few phones in any language. However, phones are more sensitive to context than words and multi-phone units such as syllable. Phone models are also considered as inadequate models for speech recognition since they do not model co-articulation effect. In other words, phone models assume that a phone in any context is considered as equivalent to the same phone in any other context – which is not true in reality (Lee, 1989).

Modeling multi-phone units like syllable is one way of modeling co-articulation effects. However, the large number (although smaller in number than words) of these units affects its trainability (Lee, 1990). Because of this, other units that take context into consideration are proposed. These are word-dependent phones and context-dependent phones or triphones. As indicated by Lee (1989), word-dependent phones are a compromise between word modeling and phone modeling. In the case of word-dependent phones, phone models are considered but they are word-dependent. This means that a phone model in one word

has different parameters from the phone model of the same phone in another word. Although word-dependent models can model context, they require large training data and storage.

Triphone or context-dependent phones are somehow similar to word-dependent phones. Triphone models are phone models that take left and right neighboring phones into consideration (Lee, 1990). Lee also indicated the fact that two phones that have the same identity and different left to right context are considered as different. Although triphone modeling is powerful since it models co-articulatory effects and insensitive to context than phone modeling, it is not trainable since there are many triphones. In addition to this, as indicated by Lee (1989), triphone modeling has two problems. One is memory wastage. The other serious problem of triphone modeling is that it assumes that every triphone context is different but in reality many phones have similar effect on their neighbors. All these subword units are considered to achieve a large vocabulary system.

Lee (1989) indicated the fact that words are the most natural unit of speech because they are what we want to recognize. Word models yield the best performance provided that sufficient training data is available. As indicated by Lee, in word models training data can not be shared between words, thus each word has to be trained individually. This means that many examples (20 or so) of each word are needed for adequate training. Although this introduces several problems in large vocabulary recognition, it brings no problem in small vocabulary. Rather, word models are suitable for small vocabulary recognition. Lee indicated the fact that successes of several small vocabulary word-based recognizers prove

the suitability of word models for small vocabulary systems. Rodman (1999) said that several words spoken together (phrases) can also be considered as a unit of recognition. When we model a phrase, the phrase is considered as one long word. Considering phrase is especially useful when we have little words that are difficult to recognize.

2.2.5. Isolated word recognition

Speech recognition systems can be categorized into discrete or isolated and continuous depending on the flow of speech. In continuous speech recognition, users generally talk in a natural way. However, it is difficult to develop than isolated word recognition because of various problems indicated in 1.1. To the contrary in isolated word recognition, users should insert artificial pauses. Although this is not a natural way of communication, isolated word recognition is easy to develop since word boundaries are clearly identified and inter-word co-articulation effects are minimized (Markowitz, 1996). Rabiner and Juang (1993) also indicated the fact that since word boundaries are clearly defined in isolated word recognition, the pattern-matching task could be reliably performed without having to be concerned about uncertainties in the endpoints of patterns being compared.

Many task-oriented isolated word recognition systems have been developed and performed adequately. Rabiner and Levinson (1990) indicated that the power of isolated word recognizer is increased substantially when designed and used to perform a particular task. Rabiner and Juang (1993) also indicated the appropriateness of isolated word recognition systems for command and control applications by saying "For many applications notably those referred to as 'command-and-control' applications, in which the user is required to

speak the command words one at a time (i.e., with distinct pauses between command words), this paradigm works well and is entirely appropriate." The term paradigm in the quotation refers to isolated word recognition.

Young et.al. (2002) discussed the problem and solution of isolated word recognition as follows.

Let each spoken word be represented by a sequence of speech vectors or observation O, defined as:

$O = o_1, o_2, \ldots, o_T$, where o_t is the speech vector observed at time t.

The isolated word recognition problem can then be regarded as that of computing

$\arg\max_i \{p(w_i/O)\}$, where w_i is i^{th} vocabulary word.

This probability is not computable directly but using Baye's Rule which is given as follows.

$$P(w_i/O) = \frac{p(O/w_i)p(w_i)}{p(O)} \qquad 2.32$$

Since P(O) is constant for a given input, the concern is on maximizing $P(O/w_i)P(w_i)$. For a given set of $P(w_i)$, the most probable word depends on only $P(O/w_i)$. However, because of the dimensionality of the speech vector O, the direct estimation of $P(o_1, o_2, \ldots / w_i)$ from examples of spoken words is not practical unless parametric model of word production such as a Markov Model is assumed. If such model is assumed, then estimation from data is possible since the problem of estimating $P(O/w_i)$ is replaced by the much simpler problem of estimating the model parameters (Ibid).

2.3. Communication

Once the recognition task is over, the recognized input should be sent to the software/hardware systems that the user need communicate with. This is a communication task of speech recognition system designed as a human-computer interface. Markowitz (1996) said that to have good speech interface there should be synchrony between the recognizer and the software/hardware systems.

Chapter Three

Application of Speech Recognition and Human-Computer Interaction Modalities

3.1. Application of Automatic Speech Recognition

Many speech recognition applications have been and are being developed. Markowitz (1996) said that "since the conclusion of the 1980's, there has been an explosion in the number and type of speech recognition applications ..." Markowitz also indicated the fact that speech recognition is mostly applied in command and control, data entry and retrieval, and dictation functions. It is important to note that one application can serve more than one functions. For example, Microsoft Office speech interface serves command and control as well as dictation functions. The following is a brief description of the above three functions.

3.1.1. Command and Control

Rodman (1999) said that the phrase command and control is borrowed from military jargon. In military, as Rodman said, the phrase refers to "central locations where officers in charge of an operation can issue command to control the movement and deployment of men and machine." In speech technology, command and control refers to verbal control of equipment or software programs (Markowitz, 1996). Rabiner and Juang (1993) also described command and control application as an application where a user speaks a single command (isolated word or phrase or connected sequence of words) and the machine acts appropriately.

Most command and control applications require small vocabulary recognition systems. Markowitz (1996) said that although the vocabulary size varies according to the operation that the equipment or the software performs, in most cases the vocabulary size ranges from ten to fifty words.

Regarding speakers model, Markowitz said that speaker dependent systems can be counter-productive for applications that are designed for ease of use since a new user should train the system before using it. Thus, speaker independent modeling is preferred.

Rabiner and Juang, (1993) said that command and control applications mostly use discrete/isolated word recognition systems. This is because a user speaks command words one at a time.

Markowitz (1996) indicated the fact that command and control applications are used in different areas. These areas include: healthcare, manufacturing, military, telecommunication, office automation, architecture, automotive, design, mining, space exploration and consumer products. In addition, it can be used to help people with disabilities. However, Markowitz indicated that command and control applications are most widely used in telecommunication and for controlling software (such as Windows and applications running under it). A description of how command and control applications are used in some of the above areas is given as follows.

In healthcare, command and control system can be used in the hospital room, the operating room, medical laboratory, etc. Patients in hospital room may use command and control system to adjust

bed, control light, door and television. In the operating room, a surgeon can, for example, use command and control system to control light, operating table and microscope (Rodman, 1999). He also said that in a hospital laboratory "where chemicals are handled and sterile conditions are required, technicians find themselves needing 'a third hand' to start an exhaust fan, turn on a light, open a door, start or stop a machine, set a timer and so forth. That third hand could be the vocal cords."

In factory environment, people are engaged in hands busy, eyes busy tasks. For example, as Rodman said, "workers in factories often have their hands full and need to use their eyes to avoid accidents." Such situations necessitate the use of speech to open and close doors, turn on and off light, adjust ventilator, etc. In addition to this, heavy machineries have risk on workers. Using a command and control system has a safety benefit to workers since it enables them to control those machines by voice without getting closer to them (Ibid).

In military, command and control systems can be used in the repair of tank engines, adjusting radio frequencies, avoiding threat and controlling aircrafts in the cockpit (Ibid).

In telecommunication, command and control system may be used in automatic call distribution, voice repertory dialer, automate call-type recognition, directory listing retrieval and credit (calling) card verification (Rabiner and Juang, 1993). For example in automatic call distribution, as Rabiner and Juang said, a command and control system can be used in place of an attendant or operator who distributes the call to the appropriate location based on user responses to the questions asked by the attendant.

Command and control systems can also be designed to serve as an interface to computer applications. Markowitz (1996) indicated the fact that such systems are often designed to enable computer novices to quickly become productive. In addition Rudnicky, Hauptmann and Lee (1993) indicated that such systems can be used in hands-busy, eyes-busy tasks. For example, using such system it is possible to change font style while typing. They also said that, it serves as a shortcut. For instance, instead of traversing many levels of hierarchy to open a file, it is possible to open it using one command "open xx" where xx is the file name.

3.1.2. Data Entry and Retrieval

Data entry is "altering the contents of a computer data file" (Rodman, 1999). It deals with supplying data to databases and other software (Markowitz, 1996), completing application forms and filling a table with numerical data (Rodman, 1999). It is a hands-busy, eyes-busy function because it allows users to continue to perform other tasks while they enter the data verbally.

Although earlier data entry systems were created for military, factories and warehouses, towards the end of 1980's it expanded to other functions and industries. Markowitz (1996) mentioned industries and functions in which data entry system is applied. These include: construction, education, engineering design, finance, healthcare, hotels, insurance, office systems, package delivery, retail and telephone repair.

Speech recognition system that is designed for data entry purpose can be small or large vocabulary, continuous or discrete, speaker dependent or independent. Earlier data entry applications use small vocabulary speech recognition systems. However, tasks that can be

performed using small vocabulary input constitute only part of potential data entry application. That means potential data entry applications require large vocabulary speech recognition systems. As the vocabulary grows, it is difficult to use speaker dependent system since each user should train the system using the vocabulary. Therefore, for large vocabulary it is better to make the system speaker independent. The choice between discrete or continuous recognition depends on other factors such as noise in the environment (Ibid)

Data access, which is also called information retrieval or information access, deals with retrieving information from a database or other file (Markowitz, 1996). Although early data access applications were telephone banking, now they are used in automotive, customer service, equipment repair, finance, law, tourism, office systems and telecommunications (Ibid).

Small vocabulary speech recognition system can be used in data access application. However, as applications incorporate extensive access of information from large databases and as they expand into full text retrieval, the vocabulary size grows. Regarding flow of speech, continuous speech in conjunction with keyword spotting is usually used. Selection of speaker model depends up on the nature and size of user population. For example, data access system designed for widespread use can be made speaker independent (Ibid).

3.1.3. Dictation System

Dictation systems "convert the stream of speech into text for letters, reports and other documents" (Markowitz, 1996). Creating a human-like listening typewriter was one hypothetical application of speech recognition which is now a reality (Rodman, 1999) because of the

technological advances in the late 1980's (Markowitz, 1996). Markowitz indicated the major areas in which dictation system are applied. These include: business, healthcare and law where extensive document-generation is needed. It can also be used by people with disabilities who have professional and/or personal dictation needs.

Earlier dictation systems were developed based on isolated word speech recognition. However, since speaking one word at a time is tedious, people prefer to type than dictate in this manner (Rodman, 1999). Nowadays, continuous speech recognition systems make dictation systems attractive (Markowitz, 1996). Although it is possible to base dictation systems on small vocabulary, speaker dependent, isolated word speech recognition, the ideal speech recognition system for dictation application is large vocabulary, speaker independent, continuous recognition system (Ibid)

3.2. Human-computer Interaction modalities

Cohen and Oviatt (1994) said that there are many alternative modalities in human-computer interaction. In this kind of interaction information can be conveyed in the form of speech, text, or visual representations (icons) (Lefebure, Duncan and Poiries, 1993). However, each mode of interaction has its own advantage and disadvantage. One mode may be good at performing one task but not for another. That means as Lefebures, Duncan, and Poirier (1993) indicated, no mode is sufficient by itself.

3.2.1. Comparison of Speech with Alternative Modalities

3.2.1.1. Graphical User Interface and Direct Manipulation

Graphical user interface (GUI) paradigm is made popular by Microsoft Windows and Apple Macintosh (Grasso, Ebert, and Finin, 1999; Cohen and Oviatt, 1994). It offers the user menus, icons, and pointing devices such as mouse as well as multiple windows used to display output. Using GUI's, users perform actions by selecting objects and then choosing the action that they want to perform from a menu instead of typing commands to perform a certain action. Many GUI's also enable users to directly manipulate graphical objects in order to perform actions on the objects they represent (Cohen and Oviatt, 1994). However, they are best used for specifying simple actions, when all references are visible and are limited in number (Grasso, Ebert and Finin, 1999). Cohen and Oviatt indicated the strengths and weaknesses of this modality as follows:

Strengths

- Direct manipulation interfaces based on familiar metaphors are intuitive and easy to use.
- Graphical user interfaces can have a consistent "look and feel" that enables users of one program to learn another program quickly.
- Menus make the available options clear, there by curtailing user errors in formulating commands and specifying their arguments.
- GUI's can shield the user from having to learn underlying computer concepts and details

Weaknesses

- The paucity of means for identifying entities. It only allows users to select currently

displayed entities. It provides users little support for identifying objects not on the screen, specifying temporal relations that denote future or past events, identifying and operating on large sets of entities, or exploiting the context of interaction.

- When numerous commands are possible, GUI's usually present a hierarchical menu structure. As the number of commands grows, the casual user may have difficulty remembering their menu location. Even if the user knows the location of the desired command, navigating the hierarchy scroll requires time and effort.
- It relies heavily on a user's hands and eyes.
- Since direct manipulation emphasizes rapid graphical response to actions, the time at which an action occurs is literally the time at which it was invoked. Thus, GUI's and direct manipulation offers little support for users who want to execute actions at some unknown but describable future time.

3.2.1.2. Speech Interaction

Cohen and Ovait (1994) indicated the fact that certain tasks would be better performed with speech. Speech interface is better at specifying complex actions when references are numerous and not visible (Grasso, Ebert and Finin, 1999). Cohen and Oviatt (1994) pointed out the strengths and weaknesses of speech interaction as follows:

Strengths

- The use of psychologically salient and mnemonic descriptions.
- Spoken language commands can shortcut the navigation of a menu hierarchy to invoke known commands.
- Ideally, spoken language systems require only a little training on the system domain.

Weaknesses

- Pure spoken language systems tend to suffer from opaque linguistic and conceptual coverage. That is the user knows that the system can not interpret every utterance but does not know precisely what it can interpret. Thus such systems can be error-prone and may lead to frustration and disillusionment.
- Many natural language expressions are ambiguous.
- Reference resolution algorithms do not always supply the correct answer because systems have underdeveloped knowledge bases and the system has little access to the discourse situation.

The above discussion on strengths and weaknesses of human-computer interaction modalities (spoken language and GUI) indicates, as Cohen and Oviatt (1994) said, the fact that the modalities have complementary advantages and disadvantages. This implies that the modalities can be integrated to create multi-modal interface that compensates the weaknesses of one modality via the strengths of the other.

3.2.2. Challenges in Designing Speech Interface

Recognition Performance

Yankelovich, Levow and Marx (1995) said that "ironically the bane of speech-driven interfaces is the very tool which makes them possible: the speech recognizers." Speech recognizers are not perfect listeners; they make mistakes (Sun Microsystem, 1998). Recognition errors are frustrating and cause the user to form incorrect conceptual model of the application's behavior (Yankelovich, Levow and Marx, 1995). Sun Microsystem (1998) indicated the fact that speech

interface designers should understand the types of recognition errors that speech recognizers make and the causes of these errors.

Yankelovich, Levow and Marx (1995) indicated possible causes of recognition errors. These include speaking before the system is ready to listen, background noise, an accent, a cold, an exaggerated tone, and speaking words that are not found in the recognizer's vocabulary (dictionary).

There are three types of recognition errors: substitution, insertion and rejection/deletion (Lee, 1989; Yankelevich, Levow and Marx, 1995). Substitution error is an error that occurs when a recognizer substitutes the user's utterance by a different legal utterance. This kind of error can be damaging. Hence in some situations designers want to explicitly verify that the user's utterance was correctly recognized (Yankelevich, Levow and Marx, 1995). Lee (1989) said that substitution error is the only possible type of error in isolated word recognition systems. Yankelevich, Levow and Marx (1995) said that rejection error is an error that occurs when the recognizer has no hypothesis about what the user said. Insertion error is an error caused when the recognizer interprets noise as a legal utterance.

Nature of Speech

Speech is transient. That means once we say it or we hear it, it is gone (Yankelovich, Levow and Marx, 1995; Sun Microsystem, 1998). Therefore, users can remember only a limited number of items in the list or words in a long sentence. In addition, users often forget the exact words they

have just spoken. This implies that in speech only application, designers should not deliver large amount of information to the user (Sun Microsystem, 1998).

Speech is also invisible. In speech only interface, lack of visual feedback can lead users to feel less in control (Yankelovich, Levow and Marx, 1995). Sun Microsystem (1998) also indicated the fact that this nature of speech makes it difficult to communicate the functional boundaries of speech-enabled applications to the user. That means it is difficult to indicate to the user what actions a speech enabled application performs, and what words or phrases users must say to perform an action.

Flexibility Vs Accuracy

Flexibility means allowing users to speak one command in different ways. Users usually require such a flexible system. But, when more flexibility is provided, more recognition errors will occur. If the recognition performance is poor, users will not accept the application. On the other hand, if the application is not flexible again users will not accept it. Thus speech interface designers should find a balance between flexibility and accuracy (performance) (Ibid).

3.2.3. Issues with Speech Input

As indicated by Laviola (1999) there are a number of issues that should be considered when dealing with speech input. Lavoila provided the following issues and their possible solutions.

- Speech direction. One practical issue with speech input is letting the computer know that the user is speaking to it or to someone else. One possible solution is to use a push-to-talk interface that helps users to provide signal to the computer when they want to speak to it.

- Microphone placement. "The microphone can be placed on the user via headset or lavalier or somewhere in the environment." Although placing microphone on the user allows for a clearer signal input and allows the user to speak at a normal or even soft volume, it creates inconvenience for users since they should wear it. Placing the microphone somewhere in the environment, on the other hand, gets the microphone off the user's body but the user have to speak at a higher volume or the microphone should be very sensitive. When the microphone is sensitive, it is more susceptible to background noise. In a noise free environment, it is preferable to use microphone that can be placed somewhere in the physical environment. However, most environments where in speech interfaces are used are not noise free. Thus, it is preferable to place the microphone on users.

- External noise. Background noise, which comes from a variety of sources, is a major problem in using speech input since it can distort the input signal and causes recognition error. We can solve this problem either by reducing external noise or by not letting the recognizer know the existence of external noise. The former is difficult since speech interfaces are mostly used in the area where people and machines are found. But, we can use the second solution which can be practicable by using noise filtering microphones.

- Recognition latency. Recognition latency represents the time lag between the input utterance and output of the recognizer. It is proportional to the size of the vocabulary. That means as the vocabulary size increases the recognition latency also increases. Recognition latency can seriously hinder the use of the system. Although it is not always

possible, the obvious means for reducing recognition latency is reducing the size of the vocabulary. Another solution is to increase the recognizer's speed by considering limited possibilities. However, this will reduce the accuracy of a recognizer. It is also possible to mask the recognition latency in some way instead of reducing it.

From the above discussion one can understand that speech recognition research has a long history (Lee, 1989). Now a days many speech recognition systems are commercially available and they have been and are being integrated with applications. However, in Ethiopia researches in the area of Amharic speech recognition started only recently. Only two master's thesis researches were conducted so far. The following section provides a review of these researches.

3.3. Amharic Speech Recognition

3.3.1. Isolated Amharic Consonant-Vowel (CV) Syllable Recognition

The first work in Amharic speech recognition was that of Solomon's. Solomon (2001) developed speaker dependent and speaker independent HMM-based prototype Amharic CV syllable recognition systems. He used Hidden Markov Model Toolkit (HTK) to develop the recognizers.

Out of 231 Amharic CV syllables he selected 41 CV syllables for the experiment purpose based on their frequency of occurrence in Amharic literature. He then recorded isolated utterances of the selected CV syllables from eight speakers who are in the age range of 20-33. Each speaker uttered 49 sets of CV syllables where each utterance consists of six

randomly generated CV syllables separated by pause. Out of 49 sets of utterances 28 sets were used to train the HMMs and the remaining 21 sets were used to test the performance of the recognizers. The recorded utterances were labeled manually.

The wave form data were parameterized into sequences of feature vectors - Mel Frequency Cepstral Coefficients (MFCCs) - which are derived from Fourier Transform based log spectra. To do this Solomon used the HTK tool HCopy.

Using the training utterances a 3 state left-to-right speaker dependent and speaker independent models were developed for each CV syllables that are included in the experiment. These models were then tested using the test data. The speaker dependent CV syllable recognizer recognized on the average 87.68% of the test data accurately. Speaker independent recognizer recognized on the average 72.75% of the test data correctly and an average of 49.21% accuracy was scored for two speakers who have no involvement in the training.

3.3.2. Sub-word based Amharic Word Recognition

Following Solomon (2001), Kinfe (2002) developed HMM-based speaker dependent and speaker independent subword-based prototype Amharic word recognizers. He considered three subword units - phoneme, triphone and consonant-vowel syllables - with the aim of identifying which subword unit is better for developing Amharic speech recognition system. Kinfe also used HTK in the course of developing the recognizers.

For the CV syllable prototype recognizer, he considered a total of 104 CV syllables. For the phoneme and triphone based models, 20 phonemes were selected out of 39 Amharic phonemes. As Kinfe said, the main criterion for selection of these units was their frequency of occurrence in Amharic speech and text. Kinfe, then selected Amharic words consisting of the selected units (CV syllables and phonemes) for training purpose. He also prepared two sets of test data: one with words randomly picked from the training data (testsetI) and the other consisting of words that are not included in the training set (testsetII).

Training and test data were recorded from 20 speakers using the HTK tool HSLab. The recorded data was parameterized using the HTK tool HCopy into Mel Frequency Cepstral Coefficients (MFCCs) which are derived from Fourier Transform based log spectra.

Using the training data Kinfe developed 3 state left-to-right speaker dependent and speaker independent phoneme models from which the triphone models have been generated. In addition, he developed 8 state left-to-right speaker dependent CV syllable models. These models were then tested using the prepared test sets.

The speaker dependent phoneme based recognizer recognized on the average 84.80% of testsetI and 83.07% of testsetII correctly. Speaker dependent triphone based recognizer recognized 84.00% of testsetI accurately and 78.00% of testsetII. The speaker dependent CV syllable recognizer recognized 74% of testsetI and 60% of testsetII correctly. Since the performance of speaker dependent CV syllable based recognizer was poor, Kinfe did not develop speaker independent CV syllable based recognizer. Thus, he developed phone

based and triphone based speaker independent recognizers. The phone based speaker independent recognizer recognized on the average 77.73% of testsetI and 75.33% of testsetII. The triphone based speaker independent recognizer has better performance than the phone based recognizer; it recognized 91.46 of testsetI and 77.87% of testsetII.

CHAPTER FOUR

Prototype Amharic Speech Interface for Microsoft Word

4.1. Introduction

As shown in fig 4.1 below, the prototype system developed in this experiment consists of two parts: Speech recognizer and communication interface. The speech recognizer accepts user's speech input and returns the transcription of a recognized word. This transcription is then input to the communication interface. The communication interface accepts transcription of the recognized word and then sends the command that the recognized word represents to the Microsoft Word. This chapter describes the development of the prototype speech recognizer and the communication interface.

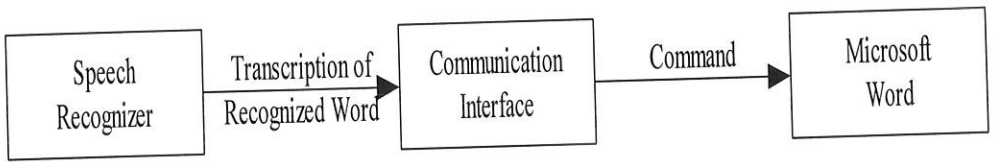

fig. 4.1. Components of the prototype system

4.2. The Speech Recognizer

As indicated in section 1.1, speech input interface requires speech recognition system. Since there is no Amharic word speech recognition system that is appropriate for this purpose, the researcher decided to develop one. Thus, a prototype HMM based speaker independent, small vocabulary, isolated Amharic word recognizer has been developed. In the course of developing the recognizer, HTK (Hidden Markov Model Toolkit) was used.

Young et. al. (2002) indicated that "HTK is toolkit for building Hidden Markov Models (HMMs)." The toolkit consists of a set of library modules and more than 20 tools (programs) written in ANSI C. It runs on UNIX and any other modern operating systems.

Although HTK can be used for other purposes (such as character recognition), it is primarily designed for building HMM based speech processing tools – particularly recognizers. As a result HTK provides much infrastructure support for speech recognition.

Young et. al. (2002) said that there are four main steps in developing a speech recognizer using HTK. Namely: data preparation, training, recognition and analysis. These four steps (which are diagrammatically shown in fig. 4.2 below), as applied to the work under discussion, are discussed below in detail. Detailed description of the HTK tools can be found in Young et. al. (2002). The History of HTK can be found from http://htk.eng.cam.ac.uk/docs/history.html.

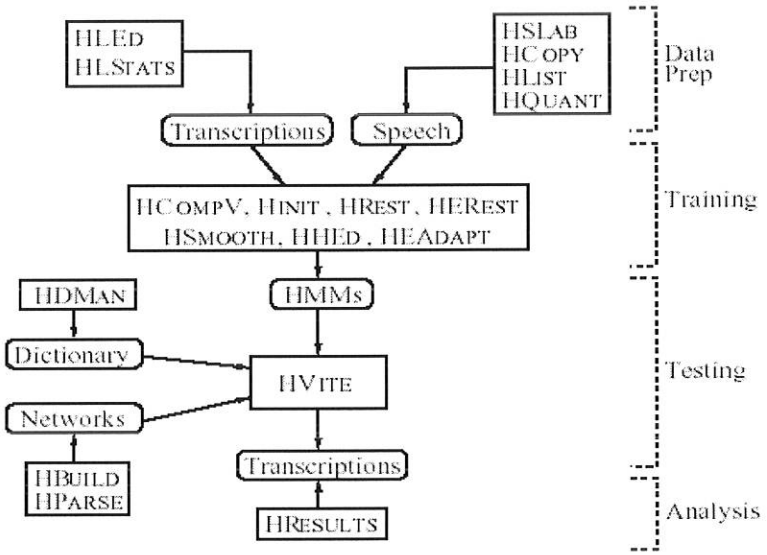

fig. 4.2. HTK processing stages (extracted from Young et. al. 2002)

4.2.1. Data Preparation

Speech data and their transcription are required for training and testing a recognizer. Since there is no such data for developing isolated Amharic word recognizer that can be used as a component in speech input interface, the researcher prepared the required data from scratch. The activities performed in this line are presented as follows.

Command Word Selection

Considering the available time and financial resource for this research, 50 words used to command and control Microsoft Word were selected, translated to Amharic and used to develop the prototype system. The 50 commands were selected from different menus (File, View, Insert, Tools, Table, Window, and Help) although the number of commands taken from these menus is different. For example more words were included from the file menu than the others since this menu includes more commonly used commands, like open (**ክፈት**), close (**ዝጋ**), new (**አዲስ**), save (**አስቀምጥ**), print (**አትም**). The menus themselves were also considered as commands. In addition, commands that are usually displayed in the toolbar, such as bold (**አድምቅ**), italic (**አዝምም**), underline (**አስምር**), align right (**ወደቀኝ**), etc. were included. List of command words used to develop the prototype system are attached as appendix A.

Data Recording

Isolated utterances of the selected Amharic commands were recorded from 26 speakers (10 female and 16 male) in the age range of 20-35. It is important to note that no analysis is made in determining the number of female and male speakers and the age range. Only availability of

speakers was considered. The recording has taken place in the faculty's computing laboratory which is not designed for speech recognition research and consequently not noise free.

The data were recorded using Praat, which is a speech analysis and syntheses program, at a sampling rate of 16 KHz. Praat is preferred over HSLab (HTK's recording tool) since the quality of speech data recorded by Praat is better than that of HSLab. The recording was done with a multimedia DELL PC which has Pentium 4 Intel processor, 1.8 G.Hz CPU speed and 128 RAM. The microphone used was headset, close-speaking, noise canceling, monophone.

The recorded data was classified into training and test sets. The speech data collected from 20 speakers (about 76.9%) was used for training purpose and the remaining data - that is collected from 6 (23.1%) speakers - was used for testing the recognizer.

Transliteration

The Amharic words should be transliterated using Roman alphabets since HTK does not accept Amharic font. In this experiment, the words were transliterated using the representation used by Solomon (2001) with some modification. For example, the word ክፈት (open) is transliterated as kIfEtI where kI represents ክ, fE represents ፈ, and tI represents ት. The transliteration of all words can be found in appendix A.

The task Grammar

Task grammar is a word level network[1] that defines all legal word sequences explicitly. Since the prototype recognizer is isolated word recognizer, it requires a simple task grammar which has the following structure:

(sil (any command word) sil)

The above structure indicates that any utterance has 3 words (considering silence as a word) in the sequence of silence, any command word and silence. The task grammar used in the prototype recognizer was defined using the HTK grammar definition format on a text editor. This format is called HParse format or EBNF (Extended Backus Naur Form) and looks like:

```
(sil
(IndEgEna_SIra | IrIdata | IyIta | adImIqI | … | yIzEtunI_asIwEgIdI | zIga)
sil)
```

where the vertical bar indicates alternatives. Full version of the task grammar is given in appendix B. fig. 4.3 shows the diagrammatical representation of the task grammar (network).

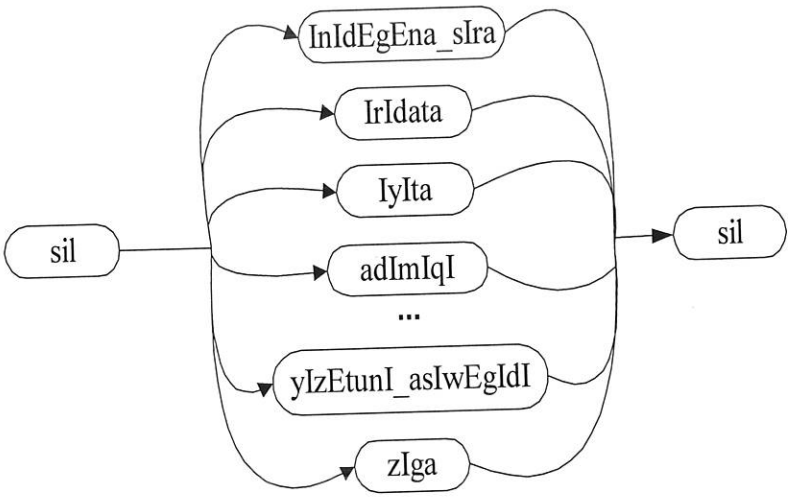

fig. 4.3 The word network

[1] A network describes the sequence of words that can be recognized (Young et. al., 2002)

However, in HTK version 3.2 (the version used in this experiment) HParse format or EBNF should be converted into an SLF (Standard Lattice Format) word network in which each word instance and each word-to-word transition is listed explicitly. The SLF can be created automatically from HParse format using the HTK tool HParse. In this experiment, the SLF was created using HParse as follows:

HParse grammar wordnet

where grammar is the file in which the HParse format is stored and wordnet is the file in which the resulting SLF is stored. The SLF is directly used by HTK recognition tool (HVite). The created SLF is given as appendix C.

The dictionary

The first step in building a dictionary is creating a sorted word list. In this experiment the word list was created manually using a text editor. For sorting the words the UNIX sort command was used.

The dictionary is then prepared for the sorted word list. HTK requires a dictionary in which each line consists of the word, the output symbol, pronunciation probability and sequence of phones or HMMs to be used in recognizing the word. The output symbol and pronunciation probability are optional. If the output symbol is not given the word itself will be output. If we don't specify the pronunciation probability, the default value (1.00) is used.

In this experiment, since words are used as a unit of recognition, one HMM was built (trained) for each word. Thus, the pronunciation dictionary has the following format.

word pronunciation

where 'word' represents the command word and 'pronunciation' represents the HMM of the word. Some of the entries in the pronunciation dictionary are shown as follows and the full version of the dictionary is given in appendix D.

InIdEgEna_sIra	InIdEgEna_sIra
IrIdata	IrIdata
IyIta	IyIta
adImIqI	adImIqI
adisI	adisI
adisI_mEsIkotI	adisI_mEsIkotI
...	
yExIhufI_saTInI_asIgEba	yExIhufI_saTInI_asIgEba
yIzEtunI_asIwEgIdI	yIzEtunI_asIwEgIdI
zIga	zIga

The Master Label file

As indicated by Young et. al (2002), many operations performed by HTK which involves speech data files assume that the speech data is divided into segments and each segment has a name or label. The set of labels associated with one speech file form a transcription. Each transcription is stored in a separate label file whose name is similar to the speech file except the extension (by default .lab is used for label file). There are two ways of providing label files to HTK tools. One way is to store different label files in one directory and specify the directory using an option provided by the tool. The other way is using MLF (Master Label File) that allows a large set of files to be stored in a single file. It can be seen as an index file that holds pointers to the label file which can be embedded in it or stored anywhere else. In this experiment MLF was used. The MLF was created using a Visual Basic program. A portion of the MLF is given below.

```
#!MLF!#
"*/tr001_1.lab"
sil
fayIlI
sil
.
"*/tr001_2.lab"
sil
```

```
adisI
sil
.
"*/tr001_3.lab"
sil
kIfEtI
sil
.
(etc)
```

The first line of the file (#!MLF!#) identifies the file as an MLF. The file name of each transcription is enclosed in double quote and the end of the transcription is indicated by dot (.). As it can be clearly understood, each speech file is segmented into 3 where the first and the last segment are labeled as sil and the second is labeled by the command word.

Coding the data

Coding a speech waveform into a sequence of parameters is one important task in the data preparation stage. It is important to note that HTK tools can also perform coding on-the-fly from the original waveform file. However, this increases the amount of preprocessing required during training. That is why coding becomes one important task in data preparation stage. In HTK, the tool HCopy is used to convert speech waveform into parameter vectors. This tool requires configuration file that specifies all the conversion parameters, and script file (file containing list of files) as an input as shown in fig. 4.4.

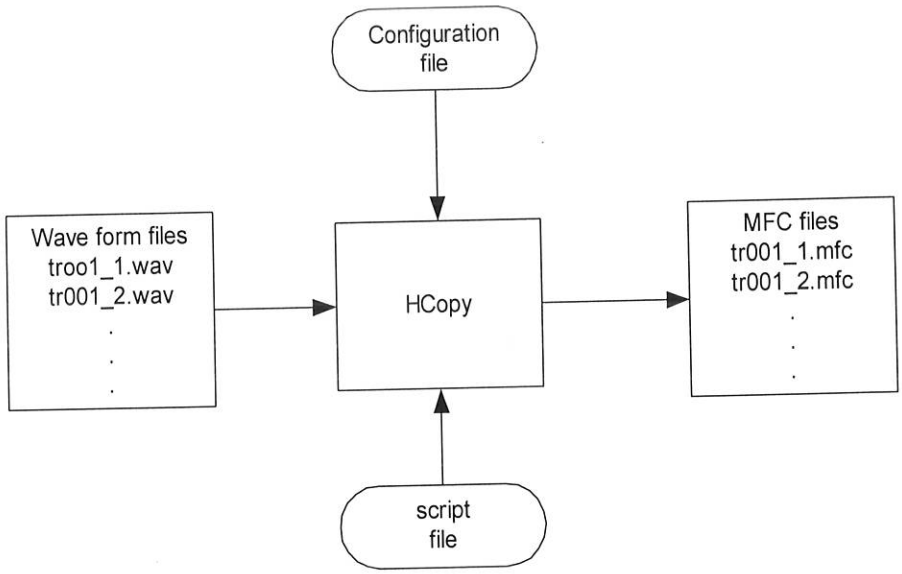

fig.4.4 Coding using HCopy (extracted from Young et. al, 2002)

Although HTK supports both LPC- and FFT-based analysis, in this study Mel Frequency Cepstral Coefficient (MFCC) derived from FFT based log spectra was used for reasons mentioned in chapter 2 section 2.1. This was done by setting the configuration variable TARGETKIND to MFCC_0_D_A. Young et. al (2002) said that adding time derivatives to the basic static parameters enhance the performance of a speech recognition system. Hence, _D (delta coefficients) and _A (acceleration coefficients) are appended.

Energy normalization is performed by HCopy by setting the variable ENORMALISE to true (which is the default setting). However, Young et. al (2002) indicated the fact that energy normalization cannot be used with live audio. Thus, since the system developed in this experiment was for live audio input, the configuration variable ENORMALISE was set to false. The configuration file used as an input to HCopy is included in appendix E.

To execute HCopy, a list of each source file and the corresponding output file is required. This list, which has the following format, was prepared using a Visual Basic program.

```
data\001\tr001_1.wav feature\tr001_1.mfc
data\001\tr001_10.wav feature\tr001_10.mfc
data\001\tr001_11.wav feature\tr001_11.mfc
data\001\tr001_12.wav feature\tr001_12.mfc
...
data\021\tr021_7.wav feature\tr021_7.mfc
data\021\tr021_8.wav feature\tr021_8.mfc
data\021\tr021_9.wav feature\tr021_9.mfc
```

where the first part lists the source files and the second lists the destination files.

HCopy was executed as follows:

```
HCopy –C hcopy.con –S wav_mfc.lst
```

where hcopy.con is the configuration file and wav_mfc.lst is the script file that lists the source file and the corresponding output files.

4.2.2. Training

Prototype definition

The first task in HMM training is defining a prototype model. The purpose of the prototype model is to define the topology of models that we want to generate and the actual numbers used in the definition are not important. Thus, the prototype defines vector size, parameter kind, number of states, allowable transitions, number of streams and mixture components in each stream.

Lee (1989) said that usually researchers choose more states than needed for the longest word and use the same model for all words. In this experiment, 17 states, left-to-right prototype model with

no skip was created using text editor and it used to define the topology of HMMs created for each word in the vocabulary. The longest word in the vocabulary consists of 15 Amharic characters and HTK requires two non-emitting states (one entry and one exit). This is why the number of states in the prototype model was defined to be 17.

In the prototype model all means were set to 0 and all variances were set to 1. Since no skip is allowed, each state has two non-zero transition probabilities. The summation of all the transition probabilities should be one for each row in the transition matrix except the last row which is 0. The non-zero transition probabilities are set to be 0.5. As indicated earlier these numbers have no effect since the purpose of the prototype is to define the model topology.

Parameter Estimation

The second step in HMM training is to estimating the parameters of the HMMs from examples of the data that they are intended to model. HTK provides four parameter estimation or training tools: HCompV, HInit, HRest, and HERest. The basic operation of these tools involves reading in a set of one or more HMM definitions and then using the speech data to estimate the parameters of these definitions.

In whole-word modeling each individual word in the vocabulary is modeled by a single HMM. The common method of building this kind of model is to use HInit to calculate initial parameters for the model and then use HRest to refine the parameters using Baum-Welch re-estimation. However, where limited training data is available and recognition in adverse noise environment is needed, fixed variance models can offer improved robustness. These fixed variance models

(generated by using the tool HCompV) are models in which all the variances are set to the global speech variance and never subsequently re-estimated. The following figure illustrates whole word training using HTK tools.

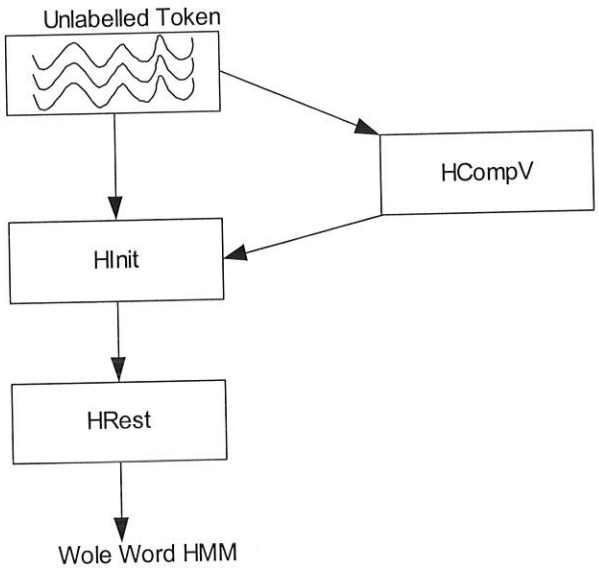

fig. 4.5. Isolated word training (extracted from Young et. al, 2002)

In this experiment, two kinds of whole word models were developed for each word in the vocabulary. First unlabeled tokens were provided to the HInit tool which initializes parameters using Viterbi training and then the resulting model was given to HRest which uses Baum-Welch re-estimation to refine the parameters of HMM. Let us call this recognizer1. Since the recognizer developed in this experiment was intended to be used in noisy environment, fixed variance model was developed. For this purpose the global variances were computed by the HTK tool HCompV. The model with the global variance was, then, used as an input to HInit and then the output of HInit was provided to HRest. In both cases (HInit and HRest) to avoid re-estimation of the global variances the option –u was used. Due to the limited number of training data, HInit could not re-estimate parameters (particularly means) for four command words – new (አዲስ), print (አትም),

print preview (የህትመት ቅድመ እይታ) and select all (ሁሉንም ምረጥ). Thus in this model these words are not included.

4.2.3. Recognition

In the previous stages of recognizer development, HMMs, dictionary and word networks were developed. In HTK, the library module Hnet takes these three elements and generates an equivalent network of HMMs which is also called recognition network[2]. This network, which is generated online at the time of recognition, is then given to Hrec (the decoding module in HTK) and used to recognize speech input.

HTK provides the tool HVite to allow the functions provided by Hnet and Hrec to be invoked from the command line. HVite is a general purpose Viterbi word recognizer. It generates recognition network and then repeatedly recognize each input utterance. The tool is useful for running experimental evaluations on test speech stored in disk files and for testing using live audio input.

The test data were given to the recognizers for recognition through the tool HVite for the evaluation purpose. Running recognizer live is one important issue of this experiment since the recognizer is intended to be used as a speech interface to Microsoft Word. The same tool (HVite) was used to run the recognizer live. However, this time a different configuration file was used since there are configuration variables such as MICIN that should be used when live recognition

[2] Recognition network consists of nodes connected by arcs. Each node is either a HMM or a word end. Since each model node is a network consisting of states connected by arcs, it can be said that a recognition network consists of HMM states connected by transitions (Young et. al, 2002).

is needed. The configuration files given to HVite in offline and live recognition are included in appendix E.

4.2.4. Recognizer Evaluation

HTK provides the tool HResults for analyzing recognizer's performance. As indicated earlier, test data was given to the recognizers and the recognized transcriptions were put in a separate MLF. Having this MLF and the MLF created in data preparation step, the performance of isolated Amharic word recognizers were evaluated by executing HResults. Both recognizers have recognized all the test speech correctly, i.e. they have 100% accuracy. The result of their live recognition presented as follows.

Five people, 3 of them were involved in the training and 2 were not, were selected based on convenience and asked to utter each Amharic command word. The recognition error was counted manually as HVite displays the recognized word for a given utterance. The following table presents the performance of the recognizers.

Table 4.1. Performance of Live Recognition

Speaker code	Recognizer1		Recognizer with fixed variance	
	Number of recognized words out of 50	percentage	Number of recognized words out of 46	percentage
trsol	50	100	39	84.8
trzeg	48	96	32	69.6
trhen	46	92	40	87
Tewol*	48	96	40	87
Temar*	47	94	33	71.7
Avg	47.8	95.6	36.8	80

* indicates people who were not involved in the training

As table 4.1 shows, Recognizer1 has good performance (above 90% for all readers including those who are not involved in the training) as compared with fixed variance model recognizer. Young et. al. (2002) said that fixed variance model is robust in noisy environment. However, as shown in table 4.1 the performance of the fixed variance model in live recognition was poorer – it recognized with a maximum accuracy of 87% - than the performance of the variable variance-based model. One possible cause for the poor performance of the fixed variance model might be the fixed nature of the variances. In such modeling the variance is set to the global variance and does not represent the variance of each model, which would have been re-estimated from the training data that each model represents.

Since the performance of Recognizer1 in live environment was better than the fixed variance model, Recognizer1 was used to build the prototype speech input interface system.

4.3. The Communication Interface

The purpose of this experiment was to develop a speech interface system that enables people pass command to the Microsoft Word orally. One component in such kind of systems is speech recognizer. The required speech recognizer (for the purpose of this experiment) was developed. After having the speech recognizer, the next task is to make the recognizer and the application communicate. Specifically, what is recognized should be sent to the application, Microsoft Word in this case, so that the application acts accordingly. A Visual Basic program was written for this purpose. The full code is attached as appendix F. This section describes how the program works.

Before presenting how the program works, it is important to describe the files used by the program. The program developed uses three files: a file that stores the recognizer's output, a file that contains Amharic words and their equivalent English command words and a file that contains the English command words and their corresponding keyboard shortcuts. The program opens the file that contains the output of the recognizer so as to get the recognized word. Then the program opens a file that stores the Amharic words and their equivalent English command words. This file can be considered as an Amharic-English dictionary developed for the purpose of this experiment. Some of the entries of this dictionary are:

fayIlI	File
adisI	New
kIfEtI	Open
zIga	Close
...	...

This dictionary is appended as appendix G. The program searches this file to identify the English command word that is equivalent with the recognized Amharic command. Some of the entries of the file that contains English command words of Microsoft Word and their keyboard shortcuts are given below. The full version of the file is appended (appendix H)

File	%f
New	%fN
Open	%fO
Close	%fC
...	...

The logic of the program is depicted by the following flow chart.

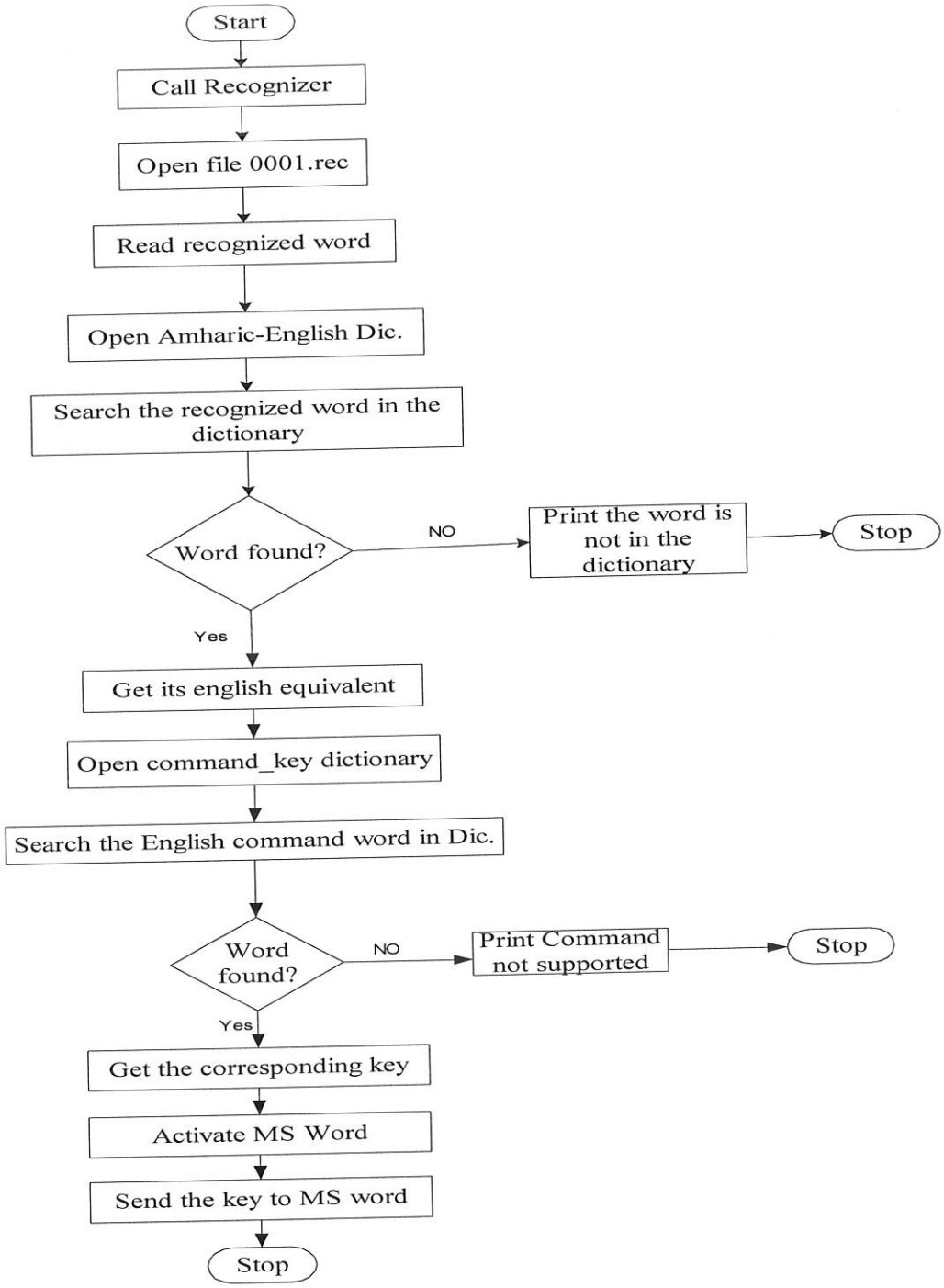

fig. 4.6. Flowchart of the communication interface

Calling the Recognizer

To call the recognizer the API (Application Programming Interface) - ShellExecute was used. First the live recognition command was written on text editor and saved as a batch file. This batch file was then executed using ShellExecute as follows.

```
ShellExecute Me.hwnd, vbNullString, "c:\marthathesis\htkworknew\finalamhrec.bat",
    vbNullString, "C:\marthathesis\htkworknew", SW_SHOWNORMAL
```

When calling the recognizer using this, two problems were encountered. After calling the recognizer the program continues executing the next statements without waiting the recognizer to finish recognition. The problem here is that before the recognizer creates the output file, the next statement tries to open the file. This problem was solved by using the following code.

```
Do While Tasks.Exists("c:\WINNT\System32\cmd.exe") = True
    DoEvents
Loop
```

The above code made the program to stop executing the other statements as long as the recognizer is running.

HVite is designed in such a way that it repeatedly accepts and recognizes input utterance until the user breaks it (using Ctrl+c in DOS environment). This is because the program has a while loop. But in the system being developed, the recognizer should accept and recognize one utterance and stop execution automatically. This was the second problem. This problem was solved by modifying the source code of HVite in such a way that it accepts and recognizes only one utterance. This is done by avoiding the while loop.

Activating Microsoft Word

In order to use the prototype system, Microsoft Word should be opened first. Although it is open, the window may not be activated. Thus activating the window is necessary since the program is designed in such a way that it sends a keyboard shortcut to the active window. To activate the Microsoft Word window the Visual Basic statement –AppActivate - was used as follows

```
AppActivate window_title
```

where window_title is the title of the window displayed in the title bar.

As it is known, users may open one file, close it, open another, etc. This has effect on the window title of Microsoft Word. That means the window title changes as the file changes. Thus the window title should be obtained automatically. For this purpose two APIs were used: FindWindow and GetWindowTitle as follows. FindWindow is used to get the window handle that is provided to the GetWindowTitle API as a parameter.

```
WinWnd = FindWindow(gcClassnameMSWord, vbNullString)
GetWindowText WinWnd, MyStr, Len(MyStr)
```

The result of GetWindowTitle is then given to the AppActivate statement.

Sending Commands to Microsoft Word

In the program, the recognizer accepts input utterance, recognizes it and puts in a file. This file is opened and then the recognized word is read and put into a string variable. This string variable is then passed to the function that opens Amharic-English dictionary and searches the string. This function returns the English equivalent of the recognized Amharic command word. The returned string is passed to the function that opens command-shortcut dictionary and searches the English command word. This function returns the corresponding keyboard shortcut. The keyboard

shortcut is then sent to Microsoft Word using the Visual Basic statement SendKeys which has the following syntax.

Sendkeys "%f", true where %f is the keyboard shortcut.

Finally the application - Microsoft Word - acts accordingly.

Evaluation of the System

Although the performance of the prototype system developed in this experiment depends on the performance of the recognizer, the system was evaluated. For this purpose based on availability 6 (3 male and 3 female) individuals were involved. Two of them were involved in the training of the recognizer and 4 were not. Considering time constraints and the longer duration required for this test, only three randomly chosen different command words were given to each individual to command and control Microsoft Word orally. While they were using the system the researcher took note of the system's performance. This evaluation was undertaken in the environment where training and test data was recorded. The system executed all the commands of the 5 individuals correctly whereas two of the commands of one individual were recognized wrongly and Microsoft Word performed wrong actions.

Finally an executable version of the program was compiled and incorporated with Microsoft Word as a Macro so that it can be executed from within the application directly.

CHAPTER FIVE

Conclusions and Recommendations

5.1. Conclusions

The purpose of this study was to explore the possibility of developing Amharic speech input interface. Towards this end, literature was reviewed on speech recognition in general and Amharic speech recognition in particular, application of speech recognition, Hidden Markov model and its application to speech recognition, HTK and human-computer interface.

Speech input interface requires speech recognition system and a communication interface. These elements were developed in this experiment for Microsoft Word. To develop the speech recognizer, speech data is required. Thus, speech data from 26 speakers (10 female and 16 male speakers) were recorded. Out of these, speech data recorded from 20 speakers were used to train recognizers and the remaining was used to test the performance of recognizers. The recorded data were coded into MFCC using the speech coding tool of HTK (Hcopy).

Using the data prepared, two HMM-based, speaker independent, small vocabulary, isolated Amharic word recognizers were developed. In both cases the topology of the HMM used was 17 states, left-to-right. The first recognizer, Recognizer1, was developed by providing unlabeled data (with label information in the form of MLF) directly to HInit - HTK's training tool - that performs re-estimation of models' parameters using Viterbi algorithm. Then, the output of HInit was provided to HRest that performs parameter re-estimation using Baum-Welch algorithm. The second recognizer was fixed variance model-based recognizer in which the global variance

(computed by HCompV) is set to each word model and never subsequently re-estimated by HInit and HRest.

The two recognizers were tested using the test data and they recognized all words in the test set correctly, i.e. both have a 100% accuracy. Since live recognition performance is important for this experiment, their live recognition performance was also tested. In live recognition Recognizer1 performed better (recognized on the average 95.6%) and thus was considered in the development of the prototype Amharic speech input interface.

Once the recognizer is developed, an effort was made to write a program that serves as a communication interface between the recognizer and the application, Microsoft Word in this case. To test the system as a whole, a total of 18 command words that are selected randomly were uttered by 6 individuals (3 commands by each) and 16 were correctly executed. Only two of the command words were recognized incorrectly resulting in Microsoft Word performing wrong actions.

In general, the study demonstrated the usability of whole word modeling for developing Amharic speech recognizers that can be used as a component of speech input interface. Although the data used in this experiment may be somewhat insufficient and the number of people whose speech was recorded is relatively limited, the result at hand shows that in live recognition the performance of the recognizer with variable variance exceeded the performance of the recognizer with fixed variance.

Moreover, based on the result obtained in this experiment, one could argue for the feasibility of the development of speech input interface to command and control application software.

5.2. Recommendations

Based on the experimental results and what has been learned throughout the course of this research, the following recommendations may be forwarded.

There is no Amharic speech corpus for building recognizers that can be used as a component of speech input interface system. Consequently this experiment was undertaken using a limited amount of speech data recorded within the time allotted for this research. This has a negative effect on the performance of the recognizer, specifically on its speaker independent feature. Thus it is important that such a speech corpus be developed. The availability of such a speech corpus will encourage researchers to conduct researches in the area. It can also facilitate the design and development of speech interface for various systems.

The current prototype system does not provide flexibility for users. That means users should know what to say in order to activate a command. Flexibility can be achieved by using keyword spotting technique. Improving the current system by using keyword spotting or any other technique is then one important action towards having a flexible speech input interface.

Although the account has demonstrated per the objective, however, the prototype system developed in this experiment does not include all command words of Microsoft Word and other applications, uses a recognizer that is not robust and does not provide any means of checking the

correctness of the recognized command word. All of which are important to have a system that can be used in real application so, further work should be done in this line. For example, in this study a push-to-talk button was used to indicate the direction of speech. But, speech interface systems that use a push-to-talk button can not be used by blind users since it is difficult for blind users to push the button whenever they want to talk to the system. Thus, in order to have a system that can be used by blind users, effort should be made to further develop the system.

This experiment explored the application of Amharic speech recognition system to command and control Microsoft Word. The general approach followed/ explored could also be applied in such related speech recognition application as dictation, data entry and retrieval.

REFERENCES

Adams, L. J. et. al. 1999. A System Design for Human Factors Studies of Speech Enabled Web Browsing. Available from: http://www.bib.ecs.soton.ac.uk/data/515/pdf/paper.pdf

Baker, James K. " Stochastic Modeling for Automatic Speech Understanding." In Reading in Speech Recognition edited by Alex Waibel and Kai-Fu Lee. pp. 297-307. California: Morgan Kuafman.

Chollet, Gerard. 1994. "Automatic Speech and Speaker Recognition: Overview, Current Issues and Perspectives." In Fundamentals of Speech Synthesis and Speech Recognition: Basic Concepts, State of the Art and Future Challenges. pp 129-147. Edited by Eric Keller. Chichester: John Willy & Sons.

Christian, Kevin, et.al. 2000. A Comparison of Voice Controlled and Mouse Controlled Web Browsing. Available from: http://citeseer.nj.nec.com

Cohen, Philip R. and Sharon L. Oviatt. 1994. The Role of Voice Input for Human-Machine Communication. Available form: http://citeseer.nj.nec.com

Furui, S. 1995. "Prospects for Spoken Dialogue Systems in a Multimodal Environment" In Proceedings of European Speech Communication Association Tutorial and research Workshop on Spoken Dialogue System: Theories and applications. pp 9-16. Vigso, Denmark.

Junqua, Jean-Claude and Jean- Paul Haton. 1996. Robustness In Automatic Speech Recognition: Fundamentals and Applications. Boston: Kluwer Academic Publishers.

Rabiner, L.R., and Juang, B-H. 1993. *Fundamentals of Speech Recognition*. New Jersey: Prentice-Hall.

Rabiner, L.R. and Stephen E. Levinson. 1990. "Isolated and Connected Word Recognition-Theory and Selected applications." In Reading in Speech Recognition edited by Alex Waibel and Kai-Fu Lee. pp. 115-153. California: Morgan Kuafman.

Rabiner, L.R. and R.W. Schafer. 1978. Digital Processing of Speech Signals. New Jersey: Prentice Hall.

Rudnicky, Alexander I. Alexander G. Hauptmann and Kai-Fu Lee. 1993. Survey of Current Speech Technology. Available from: http://citeseer.nj.nec.com.rudnicky94survey.html

Solomon Berhanu. 2001. Isolated Amharic Consonant-Vowel (CV) Syllable Recognition: An Experiment Using the Hidden Markov Model. M.Sc. Thesis, Addis Ababa University, Addis Ababa.

Sun Microsystems. 1998. Java Speech API Programmer's Guide. Available from: http://java.sun.com/products/java-media/speech/forDevelopers/jsapi-guide

Van Buskirk, R. and Mary Lalomia. 1995. A Comparison of Speech and Mouse/Keyboard GUI Navigation. Available from: http://www.acm.org/sigchi/chi95/electronic/documents/intpost/rvb-bdy1.htm

Yankelovich, Nicole, Gina-Anne Levow and Matt Marx. 1995. Designing SpeechActs: Issues in Speech User Interfaces. Available from: http://citeseer.nj.nec.com/yankelovich95designing.html

Young et. al. 2002. The HTK Book. Available from: http://htk.eng.cam.ac.uk

Appendices

Appendix A

The Command Words

English Command Word	Amharic Command Word	Transliteration
File	ፋይል	fayIlI
New	አዲስ	adisI
Open	ክፈት	kIfEtI
Close	ዝጋ	zIga
Save	አስቀምጥ	asIqEmITI
Save as	በሌላ ስም አስቀምጥ	bElela_sImI_asIqEmITI
Search	ፋይል ፈልግ	fayIlI_fElIgI
Page setup	የገጽ አቀማመጥ	yEgExI_aqEmamETI
Print preview	የህትመት ቅድመ እይታ	yEhItImEtI_qIdImE_IyIta
Print	አትም	atImI
Exit	ውጣ	wITa
Edit	አርታኣ	arItaa
Undo	ተመው	tEwEwI
Cut	ቁረጥ	qurETI
Copy	ቅዳ	qIda
Paste	ለጥፍ	lETIfI
Clear content	ይዘቱን አስወግድ	yIzEtunI_asIwEgIdI
Select All	ሁሉንም ምረጥ	hulunImI_mIrETI
View	እይታ	IyIta
Normal	ኖርማል	norImaII
Print layout	ገጽታህትመት	gExItahItImEtI
Ruler	ማስመሪያ	masImEriya
Zoom	መጠነ እይታ	mETEnE_IyIta
Insert	አስገባ	asIgEba
Insert Page numbers	የገጽ ቁጥሮች አስገባ	yEgExI_quTIrocI_asIgEba
Insert footnote	የግርጌ ማስታወሻ አስገባ	yEgIrIge_masItawESa_asIgEba
Insert Text box	የጽሁፍ ሳጥን አስገባ	yExIhufI_saTInI_asIgEba
Format	ቀይር	qEyIrI
Font	ፊደል አጣጣል	fidEII_aTaTaII
Font size	የፊደል መጠን	yEfidEII_mETEnI
Bullets and Numbering	የጠቋሚና የቁጥር ዝርዝር	yETEquamina_yEquTIrI_zIrIzIrI
Tools	መሳሪያዎች	mEsariyawocI
Spelling and Grammar	ቀለምና ሰዋስው	qElEmIna_sEwasIwI
Word count	ቃላት ቁጠር	qalatI_quTErI
Table	ሰንጠረዥ	sEnITErEZI
Insert table	ሰንጠረዥ አስገባ	sEnITErEZI_asIgEba

English Command Word	Amharic Command Word	Transliteration
Delete table	ሰንጠረዡን አስወግድ	sEnITErEZunI_asIwEgIdI
Select table	ሰንጠረዥ ምረጥ	sEnITErEZI_mIrETI
Window	መስኮት	mEsIkotI
New Window	አዲስ መስኮት	adisI_mEsIkotI
Help	እርዳታ	IrIdata
Microsoft word help	የማይክሮሶፍት ወርድ እርዳታ	yEmayIkIrosofItI_wErIdI_IrIdata
Redo	እንደገና ስራ	InIdEgEna_sIra
Bold	አድምቅ	adImIqI
Italic	አዝምም	azImImI
Underline	አስምር	asImIrI
Align right	ወደቀኝ	wEdEqENI
Align left	ወደግራ	wEdEgIra
Center	መሀል	mEhalI
Justified	ሙሉ መስምር	Mulu_mEsImErI

Appendix B

The Task Grammar

(sil(InIdEgEna_sIra | IrIdata | IyIta | adImIqI | adisI | adisI_mEsIkotI | arItaa | asIgEba | asImIrI | asIqEmITI | atImI | azImImI | bElela_sImI_asIqEmITI | fayIlI | fayIlI_fElIgI | fidElI_aTaTalI | gExItahItImEtI | hulunImI_mIrETI | kIfEtI | lETIfI | mETEnE_IyIta | mEhalI | mEsIkotI | mEsariyawocI | masImEriya | mulu_mEsImErI | norImalI | qElEmIna_sEwasIwI | qEyIrI | qIda | qalatI_quTErI | qurETI | sEnITErEZI | sEnITErEZI_asIgEba | sEnITErEZI_mIrETI | sEnITErEZunI_asIwEgIdI | tEwEwI | wEdEgIra | wEdEqENI | wITa | yETEquamina_yEquTIrI_zIrIzIrI | yEfidElI_mETEnI | yEgExI_aqEmamETI | yEgExI_quTIrocI_asIgEba | yEgIrIge_masItawESa_asIgEba | yEhItImEtI_qIdImE_IyIta | yEmayIkIrosofItI_wErIdI_IrIdata | yExIhufI_saTInI_asIgEba | yIzEtunI_asIwEgIdI | zIga)sil)

Appendix C

Standard Lattice Format

```
VERSION=1.0
N=54    L=101
I=0     W=!NULL
I=1     W=!NULL
I=2     W=sil
I=3     W=InIdEgEna_sIra
I=4     W=!NULL
I=5     W=IrIdata
I=6     W=IyIta
I=7     W=adImIqI
I=8     W=adisI
I=9     W=adisI_mEsIkotI
I=10    W=arItaa
I=11    W=asIgEba
I=12    W=asImIrI
I=13    W=asIqEmITI
I=14    W=atImI
I=15    W=azImImI
I=16    W=bElela_sImI_asIqEmITI
I=17    W=fayIlI
I=18    W=fayIlI_fElIgI
I=19    W=fidElI_aTaTalI
I=20    W=gExItahItImEtI
I=21    W=hulunImI_mIrETI
I=22    W=kIfEtI
I=23    W=lETIfI
I=24    W=mETEnE_IyIta
I=25    W=mEhalI
I=26    W=mEsIkotI
I=27    W=mEsariyawocI
I=28    W=masImEriya
I=29    W=mulu_mEsImErI
I=30    W=norImalI
I=31    W=qElEmIna_sEwasIwI
I=32    W=qEyIrI
I=33    W=qIda
I=34    W=qalatI_quTErI
I=35    W=qurETI
I=36    W=sEnITErEZI
I=37    W=sEnITErEZI_asIgEba
I=38    W=sEnITErEZI_mIrETI
I=39    W=sEnITErEZunI_asIwEgIdI
I=40    W=tEwEwI
I=41    W=wEdEgIra
I=42    W=wEdEqENI
I=43    W=wITa
I=44    W=yETEquamina_yEquTIrI_zIrIzIrI
I=45    W=yEfidElI_mETEnI
I=46    W=yEgExI_aqEmamETI
I=47    W=yEgExI_quTIrocI_asIgEba
I=48    W=yEgIrIge_masItawESa_asIgEba
I=49
W=yEmayIkIrosofItI_wErIdI_IrIdata
I=50    W=yExIhufI_saTInI_asIgEba
I=51    W=yIzEtunI_asIwEgIdI
I=52    W=zIga
I=53    W=sil
J=0     S=53    E=1
J=1     S=0     E=2
J=2     S=2     E=3
J=3     S=3     E=4
J=4     S=5     E=4
J=5     S=6     E=4
J=6     S=7     E=4
J=7     S=8     E=4
J=8     S=9     E=4
J=9     S=10    E=4
J=10    S=11    E=4
J=11    S=12    E=4
J=12    S=13    E=4
J=13    S=14    E=4
J=14    S=15    E=4
J=15    S=16    E=4
J=16    S=17    E=4
J=17    S=18    E=4
J=18    S=19    E=4
J=19    S=20    E=4
J=20    S=21    E=4
J=21    S=22    E=4
J=22    S=23    E=4
J=23    S=24    E=4
J=24    S=25    E=4
J=25    S=26    E=4
J=26    S=27    E=4
J=27    S=28    E=4
J=28    S=29    E=4
J=29    S=30    E=4
J=30    S=31    E=4
J=31    S=32    E=4
J=32    S=33    E=4
J=33    S=34    E=4
J=34    S=35    E=4
J=35    S=36    E=4
J=36    S=37    E=4
J=37    S=38    E=4
J=38    S=39    E=4
J=39    S=40    E=4
J=40    S=41    E=4
J=41    S=42    E=4
J=42    S=43    E=4
J=43    S=44    E=4
J=44    S=45    E=4
```

J=45	S=46	E=4
J=46	S=47	E=4
J=47	S=48	E=4
J=48	S=49	E=4
J=49	S=50	E=4
J=50	S=51	E=4
J=51	S=52	E=4
J=52	S=2	E=5
J=53	S=2	E=6
J=54	S=2	E=7
J=55	S=2	E=8
J=56	S=2	E=9
J=57	S=2	E=10
J=58	S=2	E=11
J=59	S=2	E=12
J=60	S=2	E=13
J=61	S=2	E=14
J=62	S=2	E=15
J=63	S=2	E=16
J=64	S=2	E=17
J=65	S=2	E=18
J=66	S=2	E=19
J=67	S=2	E=20
J=68	S=2	E=21
J=69	S=2	E=22
J=70	S=2	E=23
J=71	S=2	E=24
J=72	S=2	E=25
J=73	S=2	E=26
J=74	S=2	E=27
J=75	S=2	E=28
J=76	S=2	E=29
J=77	S=2	E=30
J=78	S=2	E=31
J=79	S=2	E=32
J=80	S=2	E=33
J=81	S=2	E=34
J=82	S=2	E=35
J=83	S=2	E=36
J=84	S=2	E=37
J=85	S=2	E=38
J=86	S=2	E=39
J=87	S=2	E=40
J=88	S=2	E=41
J=89	S=2	E=42
J=90	S=2	E=43
J=91	S=2	E=44
J=92	S=2	E=45
J=93	S=2	E=46
J=94	S=2	E=47
J=95	S=2	E=48
J=96	S=2	E=49
J=97	S=2	E=50
J=98	S=2	E=51
J=99	S=2	E=52
J=100	S=4	E=53

Appendix D

The Dictionary

InIdEgEna_sIra
IrIdata
IyIta
adImIqI
adisI
adisI_mEsIkotI
arItaa
asIgEba
asImIrI
asIqEmITI
atImI
azImImI
bElela_sImI_asIqEmITI
fayIlI
fayIlI_fElIgI
fidElI_aTaTalI
gExItahItImEtI
hulunImI_mIrETI
kIfEtI
lETIfI
mETEnE_IyIta
mEhalI
mEsIkotI
mEsariyawocI
masImEriya
mulu_mEsImErI
norImalI
qElEmIna_sEwasIwI
qEyIrI
qIda
qalatI_quTErI
qurETI
sEnITErEZI
sEnITErEZI_asIgEba
sEnITErEZI_mIrETI
sEnITErEZunI_asIwEgIdI
sil
tEwEwI
wEdEgIra
wEdEqENI
wITa

InIdEgEna_sIra
IrIdata
IyIta
adImIqI
adisI
adisI_mEsIkotI
arItaa
asIgEba
asImIrI
asIqEmITI
atImI
azImImI
bElela_sImI_asIqEmITI
fayIlI
fayIlI_fElIgI
fidElI_aTaTalI
gExItahItImEtI
hulunImI_mIrETI
kIfEtI
lETIfI
mETEnE_IyIta
mEhalI
mEsIkotI
mEsariyawocI
masImEriya
mulu_mEsImErI
norImalI
qElEmIna_sEwasIwI
qEyIrI
qIda
qalatI_quTErI
qurETI
sEnITErEZI
sEnITErEZI_asIgEba
sEnITErEZI_mIrETI
sEnITErEZunI_asIwEgIdI
sil
tEwEwI
wEdEgIra
wEdEqENI
wITa

yETEquamina_yEquTIrI_zIrIzIrI	yETEquamina_yEquTIrI_zIrIzIrI
yEfidElI_mETEnI	yEfidElI_mETEnI
yEgExI_aqEmamETI	yEgExI_aqEmamETI
yEgExI_quTIrocI_asIgEba	yEgExI_quTIrocI_asIgEba
yEgIrIge_masItawESa_asIgEba	yEgIrIge_masItawESa_asIgEba
yEhItImEtI_qIdImE_IyIta	yEhItImEtI_qIdImE_IyIta
yEmayIkIrosofItI_wErIdI_	IrIdata yEmayIkIrosofItI_wErIdI_IrIdata
yExIhufI_saTInI_asIgEba	yExIhufI_saTInI_asIgEba
yIzEtunI_asIwEgIdI	yIzEtunI_asIwEgIdI
zIga	zIga

Appendix E

Configuration Files

HCopy

```
SOURCEKIND     =    WAV
SOURCEFORMAT   =    WAV
TARGETFORMAT   =    HTK
SOURCERATE     =    625
TARGETKIND     =    MFCC_0_D_A
TARGETRATE     =    100000
SAVECOMPRESSED=     TRUE
SAVEWITHCRC    =    TRUE
WINDOWSIZE     =    250000.0
USEHAMMING     =    TRUE
PREEMCOEF      =    0.97
NUMCHANS       =    26
CEPLIFTER      =    22
NUMCEPS        =    12
ENORMALISE     =    FALSE
```

HCompV

```
TARGETFORMAT   =    HTK
TARGETKIND     =    MFCC_0_D_A
TARGETRATE     =    100000
SAVECOMPRESSED=     TRUE
SAVEWITHCRC    =    FALSE
WINDOWSIZE     =    250000.0
USEHAMMING     =    TRUE
PREEMCOEF      =    0.97
NUMCHANS       =    26
CEPLIFTER      =    22
NUMCEPS        =    12
ENORMALISE     =    FALSE
```

HInit

```
TARGETFORMAT   =    HTK
TARGETKIND     =    MFCC_0_D_A
TARGETRATE     =    100000
SAVECOMPRESSED=     TRUE
SAVEWITHCRC    =    FALSE
WINDOWSIZE     =    250000.0
USEHAMMING     =    TRUE
```

PREEMCOEF	=	0.97
NUMCHANS	=	26
CEPLIFTER	=	22
NUMCEPS	=	12
ENORMALISE	=	FALSE

HRest

TARGETKIND	=	MFCC_0_D_A
TARGETRATE	=	100000
SAVECOMPRESSED	=	TRUE
SAVEWITHCRC	=	FALSE
WINDOWSIZE	=	250000.0
USEHAMMING	=	TRUE
PREEMCOEF	=	0.97
NUMCHANS	=	26
CEPLIFTER	=	22
NUMCEPS	=	12
ENORMALISE	=	FALSE

Hvite

TARGETKIND	=	MFCC_0_D_A
TARGETRATE	=	100000
SAVECOMPRESSED	=	TRUE
SAVEWITHCRC	=	FALSE
WINDOWSIZE	=	250000.0
USEHAMMING	=	TRUE
PREEMCOEF	=	0.97
NUMCHANS	=	26
CEPLIFTER	=	22
NUMCEPS	=	12
ENORMALISE	=	FALSE
SOURCEKIND	=	HAUDIO
SOURCEFORMAT	=	WAV
USESILDET	=	TRUE
MEASURESIL	=	FALSE
OUTSILWARN	=	TRUE
MICIN	=	TRUE
SOURCERATE	=	625
TARGETFORMAT	=	HTK

HVite (Live)

TARGETKIND	=	MFCC_0_D_A
TARGETRATE	=	100000
SAVECOMPRESSED	=	TRUE
SAVEWITHCRC	=	FALSE
WINDOWSIZE	=	250000.0
USEHAMMING	=	TRUE
PREEMCOEF	=	0.97
NUMCHANS	=	26
CEPLIFTER	=	22
NUMCEPS	=	12
ENORMALISE	=	FALSE
SOURCEKIND	=	HAUDIO
SOURCEFORMAT	=	WAV
USESILDET	=	TRUE
MEASURESIL	=	FALSE
OUTSILWARN	=	TRUE
MICIN	=	TRUE

Appendix F

The Program

```
Option Explicit
Private Declare Function GetWindowText Lib "user32" Alias "GetWindowTextA"
(ByVal hwnd As Long, ByVal lpString As String, ByVal cch As Long) As Long
Private Declare Function FindWindow Lib "user32" Alias "FindWindowA" (ByVal
lpClassName As String, ByVal lpWindowName As String) As Long
Private Declare Function GetWindowTextLength Lib "user32" Alias
"GetWindowTextLengthA" (ByVal hwnd As Long) As Long
Private Declare Function ShellExecute Lib "shell32.dll" Alias "ShellExecuteA"
(ByVal hwnd As Long, ByVal lpOperation As String, ByVal lpFile As String,
ByVal lpParameters As String, ByVal lpDirectory As String, ByVal nShowCmd As
Long) As Long

Const SW_SHOWNORMAL = 1
Const gcClassnameMSWord = "OpusApp"
Private Sub Form_Load()

    Dim MyStr As String
    Dim WinWnd As Long
    Dim readfile As String
    Dim recfile As File
    Dim mystream As TextStream
    Dim myfileobject As New FileSystemObject
    Dim x, i, key As String

    ShellExecute Me.hwnd, vbNullString, "c:\htkworknew\live.bat",
vbNullString, "C:\htkworknew", SW_SHOWNORMAL

    Do While Tasks.Exists("c:\WINDOWS\System32\cmd.exe") = True
        DoEvents
    Loop

    For i = 1 To 1000: Next i

    WinWnd = FindWindow(gcClassnameMSWord, vbNullString)
    MyStr = String(GetWindowTextLength(WinWnd) + 1, Chr$(0))
    GetWindowText WinWnd, MyStr, Len(MyStr)

    Set recfile = myfileobject.GetFile("c:\htkworknew\0001.rec")
    Set mystream = recfile.OpenAsTextStream(ForReading)
    Do While (mystream.AtEndOfStream = False)
        readfile = mystream.ReadLine
        readfile = readfile & ""
        If readfile <> "sil" Then
          Exit Do
        End If
    Loop

    readfile = identifyword(readfile)

    If readfile <> "" Then
        key = identifykey(readfile)
```

```
        Else
            x = MsgBox("The Word is not recognized", vbOKOnly)
        End If

        If key <> "" Then
            AppActivate MyStr
            SendKeys key, True
        Else
            x = MsgBox("This command is not supported", vbOKOnly)
        End If

    End Sub

    Private Function identifyword(ByVal str) As String
        Dim amhaeng As File
        Dim mystream As TextStream
        Dim myfileobject As New FileSystemObject
        Dim readfile As String
        Dim strpos, strpos1
        Dim strlen As Integer

        Set amhaeng = myfileobject.GetFile("c:\htkworknew\amha_eng.txt")
        Set mystream = amhaeng.OpenAsTextStream(ForReading)

        Do While (mystream.AtEndOfStream = False)
            readfile = mystream.ReadLine
            strpos = InStr(1, readfile, str)
            If strpos > 0 Then
                strpos1 = InStr(1, readfile, Space(1))
                If strpos1 > 0 Then
                    strlen = Len(readfile)
                    identifyword = Right(readfile, strlen - strpos1)
                    Exit Do
                End If
            End If
        Loop

    End Function

    Private Function identifykey(ByVal engword) As String
        Dim engkey As File
        Dim mystream As TextStream
        Dim myfileobject As New FileSystemObject
        Dim readfile As String
        Dim strpos, strpos1
        Dim strlen As Integer

        Set engkey = myfileobject.GetFile("c:\htkworknew\word_key.txt")
        Set mystream = engkey.OpenAsTextStream(ForReading)

        Do While (mystream.AtEndOfStream = False)
            readfile = mystream.ReadLine
            strlen = Len(readfile)
            strpos = InStr(1, readfile, engword)
            If strpos > 0 Then
                strpos1 = InStr(1, readfile, "%")
                If strpos1 > 0 Then
```

```
                identifykey = Right(readfile, (strlen - strpos1) + 1)
                Exit Do
            Else
               strpos1 = InStr(1, readfile, "^")
               If strpos1 <> 0 Then
                  identifykey = Right(readfile, (strlen - strpos1) + 1)
                    Exit Do
               End If
            End If
        End If
    Loop

End Function
```

Declaration

This thesis is my original work and has not been submitted as a partial requirement for a degree in any other university.

Martha Yifiru
July, 2003

The thesis has been submitted for examination with our approval as university advisors.

Tesfaye Biru July, 2003	Solomon Berehanu July, 2003	Kinfe Tadesse July, 2003