ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY
SCHOOL OF ENGINEERING DEPARTMENT OF COMPUTING

Large Vocabulary Spontaneous Speech Recognition For Tigrigna

By
Ataklti Kahsu Gebremariam

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

September 2016

ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY

SCHOOL OF ENGINEERING DEPARTMENT OF COMPUTING

Large Vocabulary Spontaneous Speech Recognition For Tigrigna

By

Ataklti Kahsu Gebremariam

September, 2016

Name and signature of members of the examining board

Name	Signature	Date
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____
_____	_____	_____

Acknowledgements

First I would like to thank the almighty GOD for giving me the strength and making me have a set of distinctive goals in my life.

I am also grateful to my advisor Dr. Solomon Teferra and my co-advisor Hafte Abera (candidate PHD) for their continuous support and constructive comments until the completion of the work.

My thanks also go to Adugna D. and Bantegize A. encouragement and support during the whole work and whom also gave me insight to this this work.

My appreciation also go to all Adama Science and Technology University community and to my colleagues and friends for the brotherly love shown me during my period of study in Adama Science and Technology University.

Finally, I would like to thank my wife Mrs. Senait Tsehaye, my friend Tekle Gebreyohans and My brother Hagazi weldu for being by my side in all my ups and downs and Mrs. Helen G/her for checking this thesis for grammatical and spelling errors and providing me important comments.

Dedication

Dedicated to all of my friends who helped me. . .

Table of Contents

List of Tables

List of Figures

List of Appendices

Acronyms

ANN	Artificial Neural Networks
ASR	Automatic Speech Recognition
CMU	Carnegie Mellon University
DWT	Dimtsi Woyan Tigray
FBC	Fana Broadcasting Corporation
HMM	Hidden Markov Model
HTK	Hidden Markov Model Toolkit
MFCC	Mel Frequency Cepstral Coefficient

Abstract

Speech is the most natural form of human communication and speech processing has been one of the most exciting areas of the signal processing. Speech recognition technology has made it possible for computer to follow human voice commands and understand human languages.

Tigrigna is a widely spoken languages in Northern part of Ethiopia and an official language of Eritrea. In spite of its importance, research on Tigrigna Speech Recognition (SR) is unfortunately still very limited.

This thesis proposes and describes a research attempt on designing and developing a speaker-independent Spontaneous automatic Speech Recognition System for Tigrigna. The acoustic model of the Speech Recognition System is developed using the Carnegie Mellon University's ASR development tool (Sphinx) while SRIM tool is used for the development of the language model.

The system uses a database containing 3,524 sentences and phonetic dictionary containing about 10731 words and their pronunciations among these 349 sentence were used for testing the performance of the system. For the Language Model 30937 sentences have been used to develop a trigram language that contains 57,847 unigram, 292554 bi-grams and 74,178 tri-grams.

Performance tests were then conducted at various stages using test data and finally, 36.83% word level accuracy and 13.67% sentence level accuracy were obtained. The results are encouraging and with more optimization works better results can be achieved. Finally, conclusions were drawn and recommendations were made in line with the analysis and findings

CHAPTER ONE

1. INTRODUCTION

This chapter initially explains the general introduction to Automatic Speech Recognition. Then, it presents the overall objective, methodology, significance of the study, scope and limitation of the thesis and problem that has to be addressed finally how the thesis organized will be presented.

1.1 Background

1.1.1 Overview of Automatic Speech Recognition (ASR)

Information processing machines have become ubiquitous. However, the current modes of human machine communication are geared more towards living with the limitations of computer input/output devices rather than the convenience of humans. Speech is the primary mode of communication among human beings. On the other hand, prevalent means of input to computers is through a keyboard or a mouse. It would be nice if computers could listen to human speech and carry out their commands [1].

Speech recognition is a field of computer science that deals with designing computer systems that recognize spoken words. It is a technology that allows a computer to identify the words that a person speaks through a microphone or tele- phone. Speech recognition can be defined as the process of converting an acoustic signal captured by a microphone or a telephone, to a set of words [2] [3]. ASR is one of the fastest developing fields in the framework of speech science and engineering. As the new generation of computing technology, it comes the next major innovation in man-machine interaction, after functionality of TTS, supporting IVR systems.

The first attempts (during the 1950s) to develop techniques in ASR, which were based on the direct conversion of speech signal into a sequence of phoneme like units, failed [4]. The first positive results of spoken word recognition came into existence in the 1970s, when general pattern matching techniques were introduced. As the extension of their applications was limited, the statistical approach to ASR started to be researched, at the same period. Nowadays, the statistical techniques succeed over ASR applications.

Common Speech Recognition Systems these days can recognize thousands of words. The last decade has witnessed dramatic improvement in speech recognition technology, to the extent

that high performance algorithms and systems are becoming available. In some cases, the transition from laboratory demonstration to commercial deployment has already begun [3].The reason for the evolution of ASR, hence improved is due to its applications in many aspects of our Command recognition, information inquiry, dictation, personal computer interfaces, and automated telephone services, interactive voice response, and special purpose commercial and industrial systems are some of the application areas of ASR. It can also be used in education for language learning. It has other important applications for handicapped people on helping them with their daily life and their communication with the rest of the society.

In recent years, speech recognition technology has advanced to the point where it is used by millions of individuals to automatically create documents from dictation. Medical transcriptions listen to dictate recordings made by physicians and other health care professionals and transcribe them into medical reports, correspondence and other administrative material.

An increasingly popular method utilizes speech recognition technology, which electronically translates sound into text and creates transcripts and drafts of reports. Transcripts and reports are then formatted, edited for mistakes in translation, punctuation or grammar and checked for consistency and any possible errors. Transcriptions working in areas with standardized terminology, such as radiology or pathology, are more likely to encounter speech recognition technology.

1.2 Typical ASR Procedures

Figure 1.1 illustrates the major processes of ASR systems and their relation to applications. In feature extraction, capture of the pressure waves as a series of samples is performed and signal processing techniques are applied to the speech signal in order to extract the features that distinguish different phonemes from each other. End pointing is necessary to identify where speech is present in a captured signal. Given the features extracted from the speech, Acoustic Modeling provides probabilities for different phonemes at different [1] time instants. Language Modeling, on the other hand, defines what kind of phoneme and word

Sequences are possible in the target language or application at hand, and what their probabilities are. The Acoustic Models and Language Models are used in decoding for searching the recognition hypothesis that fits best to the models. Recognition output can then be used in various applications [5].

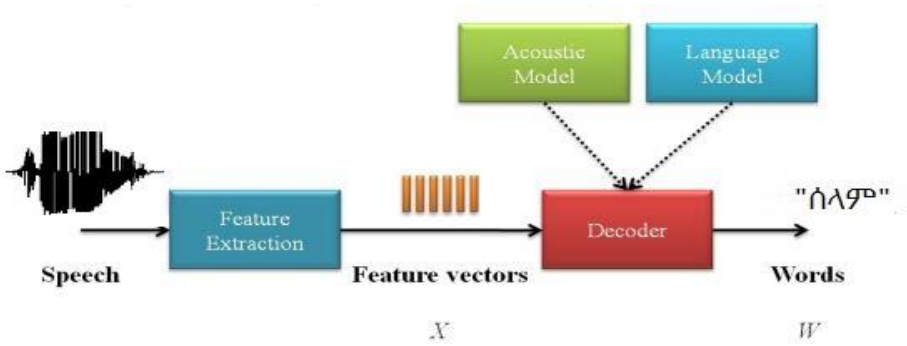

Figure1.1 *Automatic Speech Recognition process [6]*

Classification of Automatic Speech Recognition

Speech Recognition Systems can be separated in several different classes by describing what types of utterances they have the ability to recognize. Based on one of the difficulties of ASR is the ability to determine when a speaker starts and finishes an utterance. Most ASR packages can fit into more than one class. Depending on which mode they're using, Speech Recognition Systems can be categorized into different groups depending on the constraints imposed on the nature of the input speech [5].

Number of speakers: - A system is said to be speaker independent if it can recognize speech of any and every speaker; such a system has to learn the characteristics of varied and large number of speakers. A large amount of users' speech data is necessary for training a speaker dependent system. Such a system does not recognize others' speech well. Speaker adaptive systems, on the other hand, are speaker independent systems to start with, but have the capability to adapt to the voice of a new speaker provided that sufficient amount of his/her speech is provided for training the system.

Vocabulary size: - An ASR system that can recognize a small number of words is called a small vocabulary system. Medium vocabulary systems can recognize a few hundreds of words. Large and Very Large ASR systems are trained with several thousands and several tens of thousands of words respectively *[7]* Examples of application domains of small, medium and very large vocabulary systems are telephone/credit card number recognition, command and control, dictation systems respectively.

Spectral bandwidth (Channel type):- The characteristics of the channel can affect the speech signal. It may range from telephone channels (with a bandwidth about 3.4 kHz) to wireless channels with fading and with a sophisticated voice *[6]* such a speech is called narrow-band speech. In contrast, normal speech that does not go through such channel is call wide band speech; it contains a wider spectrum limited only by the sampling frequency. As a result, recognition accuracy of ASR systems trained with wide band speech is better. Moreover, an ASR system trained with narrow band speech performs poorly with wide band speech and vice versa.

Nature of the utterance:-A user is required to utter words with clear pause between words in an Isolated Word Recognition System. A Connected Word Recognition system can recognize words, drawn from a small set, spoken without need for a pause between words. On the other hand, Continuous Speech Recognition Systems recognize sentences operated on speech in which words are connected together that is separated by pause. Spontaneous speech, as opposed to planned speech, is a more natural way in which people communicate with each other *[8] [7]*. However, the recognition of spontaneous speech is made more challenging by the severe pronunciation variants and unpredictable pauses or laughter in between words. Spontaneous Speech Recognition System can handle such disfluencies events such as ah, am or false starts, grammatical errors present in a conversational speech *[9] [10]*. Based on this categorization, this research was focused on, Speaker independent Spontaneous speech, with large vocabulary size.

1.3 Statement of the Problem and Justification

Advances in electronic and computer technology are causing an explosive growth in the use of machines for processing information. In most cases this information originates from a human being and is ultimately used by a human being. There is thus a need for effective ways of transferring information between people and machines, in both directions. One very convenient way in many cases is in the form of speech, because speech is the communication method most widely used between humans; it is therefore extremely natural and requires no special training [11].

Speech recognition is a rich field of research for both practical and intellectual reasons. With this regard, the use of speech to communicate between human beings is inevitable of the primary choice and with no substitutes. This is because there are physically disabled people,

people, which means handicapped people, who use the language but one or the other way lacked the means to interact with the machine. For instance, people with difficulty hearing could use a system connected to their telephone to convert the caller's speech to text. In addition to this the domain is rich in variety of application such as dictation, command and control, telephony [12].

Moreover, some researches on speech technology in Ethiopian languages in general have been conducted by different researchers. [13] And [8] made efforts to develop Text-to-Speech Synthesis of the Amharic Language and Automatic Speech Recognizer for Afaan Oromo, respectively. [7] Hidden Markov Model Based Large Vocabulary, Speaker Independent Continuous Tigrigna Speech Recognition and also [14] developed sub word-based isolated word recognizers. These works developed speaker dependent and speaker independent systems using HMM. In recent years Amharic speech technology stepped towards spontaneous speech recognition; developed by [9] speaker independent Spontaneous Speech Recognition Systems using HMM as well as [10] BRANA (ብራና): application of Amharic Speech Recognition System for dictation in judicial domain.

Tigrigna is an agglutinative language and is highly productive in terms of word generation. For instance, from a verbal stem one can generate hundreds of distinct words by concatenating various suffixes, each of them including different meanings. Taking into account the term "spontaneous", as the knowledge of the researcher, there is not a system developed for Tigrigna, which processes naturally spoken utterances, except, a research made by [7] on Hidden Markov Model Based Large Vocabulary, Speaker Independent Continuous Tigrigna Speech Recognition using HTK environment. However, recognizing human speech, specifically spontaneous speech is necessary to increase the way human and computer interacts for the full working recognizer in the language. Spontaneous speech is most convenient way because it is the communication method most widely used between humans [12].

The purpose of this thesis therefore is to design and train a speaker independent Spontaneous Speech Recognition Systems that could be used by application developers to develop application that will take Tigrigna language speakers aboard to the current information and communication technologies to fast-track the benefits of speech. To this end, this study attempts to explore and answer the following research questions.

- How to deal with challenges facing on developing Tigrigna Spontaneous Speech Recognition System?
- How the system operates on Context Dependent (CD) and Context Independent(ID)
- What are the effects of modeling non-speech events on speech recognizer performance?

1.4 Objective of the Study

1.5.1 General Objective

The general objective of this research is to explore the possibility of developing speaker independent large vocabulary Spontaneous Speech Recognizer for Tigrigna language.

1.5.2 Specific Objective

In order to achieve the general objective, a set of specific objectives are set. These include:

- Perform a comprehensive literature review of both related to automatic speech recognizer and the language under study.
- Explore speech recognition tools and techniques that are appropriate for this research.
- Collect sample data and segmenting into a sentence level, digitize, and perform feature extraction with the appropriate tools.
- Developing spontaneous speech corpus from the collected data for training and testing.
- Designing prototype Tigrigna language spontaneous speech recognition sys- tem using selected model.
- Evaluate the performance of the recognizer using the test corpus.
- Analyze results and forward conclusion and recommendations for further research in the area.

1.5 Significance of the Study

Recent progress ASR technology has enabled the development and deployment of more sophisticated and more accurate speech recognition applications. This progress, combined with an explosion in the capabilities and use of computing devices, makes it feasible for ASR to become a common feature and service for current technology. In addition to this, making communication between human and machine through speech like humans communication

with each other that is through spontaneous speech is most convenient way because it is the communication method most widely used between humans [12].

Many efforts had been made towards Tigrigna speech recognition to help handicapped and blind Tigrigna speakers who speak clearly. After the completion of this research, it will provide a valuable model for spontaneous speech recognizer. The findings of this research.

- Supports handicapped and blind Tigrigna speakers to convert their speech to text.
- Extends the general application of ASR to Tigrigna language on command and control, telephony and use applications of automatic speech recognizer for disabled people with their own language.
- Develops Spontaneous speech recognizer that creates situation in identifying the possible way of developing an automatic speech recognizer for Tigrigna language
- It also improves the human computer interaction particularly to the speaker of Tigrigna.
- Can also serve as a baseline reference and initiative to conduct further studies in the future.

1.6 Scope and Limitation of the Study

Language Model, Acoustic Model and pronunciation dictionary are the main part of the recognizer. Due to limitation of time and the research, the researcher is limited in working with 24 randomly selected interviews and discussions out of which 12 are female speakers and the other 12 are male speakers.

In this work dialect and accent were not considered within the language under investigation, because the variation in dialect and accent causes difference in the performance of the recognizer and need to perform a lot of acoustic tasks.

Finally on this work speaker independent Spontaneous speech recognition for Tigrigna language with large size (4 hours and 45 minutes of training speech) of corpus have been developed, using conversational speech.

1.7 Methodology

To achieve the objective, the applied methodology of the research follows the following standards. In the following, literature reviews, data collection and selection, testing and training and evaluations method used are detailed.

1.7.1 Literature Review

An Extensive literature review was conducted in order to investigate the fundamental principles of various approaches, techniques, algorithms and tools that would be best for this research. Moreover Hidden Markov Model and its application in speech and researches conducted on Ethiopian language speech recognition was also part of the review. To understand how the CMU Sphinx works, CMU Sphinx book and various and domain experts were consulted. Finally reviews on the Tigrigna language's phonetic characteristics also assessed so as to have general understanding on the problem domain.

1.7.2 Data Collection and Preparation Methods

In order to generate reliable statistics for both acoustic and Language Models, large amounts of data are required. It follows that a training corpus should be sufficiently large, consisting of speech samples spoken by men and women on different issues. Collecting such data set obviously is a major task of this thesis involving tasks like selection of data source.

Audio data was collected and converted to .wav file extension using .wav converter tools, those data were not restricted to any domain rather they were general, such as an interview made between two or among many persons on different issues (domains) like sport, entertainment, politics, economy, health matter and others. Finally the data were portioned in sentence level and speech segments with non-speech segments such as music or noise had been filtered out and modeled.

Many languages have speech corpora that are commercially available or developed by other researchers like Amharic developed by [15]. Tigrigna have not any speech corpora either commercial or developed by other researchers. For this thesis a database having 4 hours and 45 minutes of training and 30 minutes of test speech had been prepared.

The process of learning about the sound units is called training. The process of using the knowledge acquired to deduce the most probable sequence of units in a given signal is called decoding, or simply recognition.

1.7.3 Training and Test Sets

The process of learning about the sound units is called training. The process of using the knowledge acquired to deduce the most probable sequence of units in a given signal is called decoding, or simply recognition. The training sets contain the majority of the speech corpora. For the development of the training corpus 24 speakers with 12 female and 12 male speakers with in different agenda were used. The definition of speaker independent, test sets as fundamental requirement is that all test speakers must have been unseen for all of the training sets 4 of new (unseen) speakers 2 of them were female and 2 of them were male speakers were used for testing purposes.

1.7.4 Modeling Technique

Hidden Markov Model (HMM) was used to build the recognizer. HMM is a powerful technique capable of robust modeling of speech and it is a parametric model that is particularly suitable for describing speech events. HMMs have two stochastic processes which enables the modeling not only acoustic phenomena but also of timescale distortion.

Furthermore, efficient algorithms exist for accurate estimation of HMM parameters which will be discussed on Chapter three. HMMs are succinct representation of speech events therefore they require less storage than many other strategies. HMM is the most widely applied and said most successful speech modeling technique in spontaneous speech recognizer. [9] And [10] used HMM for spontaneous speech recognizer. As well as [4] also applied HMMs for Tigrigna language on developing Large Vocabulary, Speaker Independent Continuous Tigrigna Speech Recognizer.

For developing purpose Sphinx-4 Speech Recognition System was used which is built using Hidden Markov Model that became the predominant technique for speech recognition It has been built entirely in the Java programming language. This is highly modular and flexible, supporting all types of HMM-based Acoustic Models, all standard types of Language Models generated using SRILM, Unicode and multiple search strategies [16]. To develop the required front end, java programming language was also used. Since Java is rich in Application programming Interface (API), platform independent and Sphinx-4 decoder is developed with java programming.

1.7.5 Evaluation of Performance and Testing Techniques

Evaluation is perhaps a more objective way to carry out evaluation of speech systems. Usually, a set of testing data is first recorded, then the sentence error rate (SER) or word error rate (WER) are then found. Sentence error rate are defined as the percentage of the number of misrecognized sentence out of all tested utterances. Whereas WER is usually defined as [17]

$$A = \frac{H-S-I-D}{H} \times 100 \quad (1.1)$$

Where: - H is the total number of words in the reference, S is the number of substitution errors, I is the number of insertions errors and D is the number of deletions errors.

1.8 Organization

This paper is organized into six chapters. The first chapter was a background information about ASR, Typical ASR procedures and Types of ASR. It was also justified the need for the study, presented the objectives of the study and also discussed the methodology followed throughout the work.

Second chapter will focuses on literature reviews of Basics of Speech Recognition, process of Speech perception and production in human beings and how they are represented in digital signal processing and related works on the development of Tigrigna language speech recognizers and on other languages will be discussed.

The third chapter introduces Tigrigna phonetics and writing systems. It also deliberates on the fundamentals of Tigrigna phonetics place and manner of articulation for Tigrigna language.

The fourth chapter deals with the basics of HMMs and provides an overview of CMU Sphinx and the accompanying used tools based on the CMU Sphinx.

The fifth chapter discusses the design and implementation details of the Tigrigna Speech Recognition System and the analysis made based on the findings and the last chapter presents the conclusions drawn and the recommendations made.

CHAPTER TWO

2. REVILITERATURE REVIEW

This chapter is intended to present speech recognition concepts and how speech signal is produced and processed. It start with a brief introduction into the acoustics and the goals of feature extraction methods in general. Then, describe the feature extraction techniques for Mel frequency.

2.1 Basics of Speech Recognition

2.1.1 Introduction

It is obvious that human communication ranges from the process of speech formulation to speech perception between the talker and listener with five different elements. This is necessary to understand how the speech signal is produced and perceived by human beings and how this speech signal is being processed in computer understandable. Section 2.2will carefully examines these five elements human communication process and the human Speech Production Systems.

Understanding fundamental concepts to the field of speech recognition and its process is an essential subject that has to be considered before one can pursue and decide which approach to use for speech recognition. Along with relevant background to the field of Speech Recognition System HMM based recognizers has vital role on modeling Speech Recognition Systems. Section 2.3 is intended to provide theoretical background of speech recognition with its Dimensions.

Finally, it is useful to understand the principals behind the development of ASR system to in order to achieve well performance system review of related works has great roll the last section intended to explain the speech recognition on what other did and how they did it with and other issues will be discussed.

2.2 Speech Perception and Production in Human Beings

The production of (speech generation) begins when the talker formulates (in his/her mind) what he/she wants to transmit to listener via speech. As shown in Figure 2.1. Speech formulation,

Human vocal mechanism, Acoustic air, Perception of the ear and speech comprehension are the five elements of human communication process.

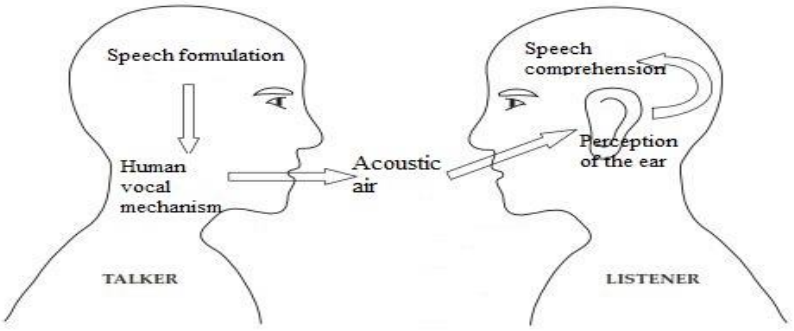

Figure 2.1: *Human speech perception System [5].*

Speech formulation is associated with the generation of the speech signal in the talker's mind. The formulated is used by the human vocal mechanism to produce the actual speech waveform. The waveform is then traveled via the air called Acoustic air to the listener. During this transfer, the acoustic wave form can be affected by external sources, such as noise, resulting in a more complex waveform. When the wave reaches the listener's hearing system (the ears), the listener perceives the waveform referred as perception of the ear and the listener's mind speech comprehension starts processing this waveform to comprehend its content so the listener understands what the talker is trying to convey. One issue with speech recognition is to "simulate" how the listener processes the speech produced by the talker. There are several actions taking place in the listeners head and hearing system during the process of speech signals perception. The perception process can be seen as the inverse of the speech production process [5].

The most important parts of the human vocal mechanism are the vocal tract together with nasal cavity, which begins at the velum. The velum is a trapdoor- like mechanism that is used to formulate nasal sounds when needed. When the velum is lowered, the nasal cavity is coupled together with the vocal tract to formulate the desired speech signal. The cross-sectional area of the vocal tract is limited by the tongue, lips, jaw and velum and varies from 0-20 cm^2 [5].

When humans produce speech, air is expelled from the lungs through the trachea. The air owing from the lungs causes the vocal cords to vibrate and by forming the vocal tract, lips, tongue, jaw and maybe using the nasal cavity, different sounds can be produced. This human speech production system can be modeled into linear filter exited by an impulse train which is

voiced and white noise which is also unvoiced. The simplified model of speech production according to [5] can be drawn in Figure 2.2 and the model works as follows.

This perception process is modeled by the signal processing and the speech decoder components of the speech recognizer, whose aim is to process and decode the acoustic signal X into a spectrum $S(f)$

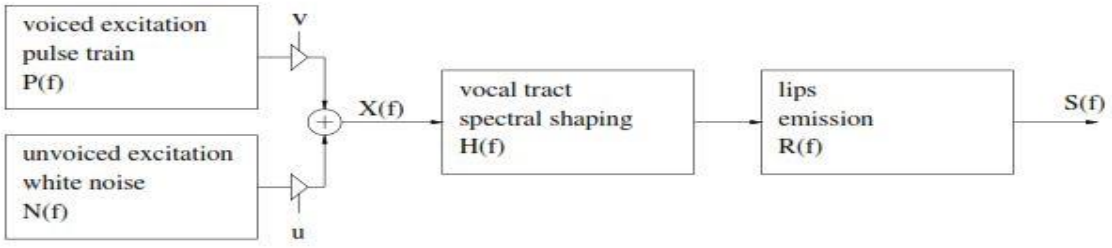

Figure 2.2*: *A simple model of speech production [18]

Voiced exiting is modelled by a pulse generator which generates a pulse train (of triangle shaped pulses) with its spectrum given by P (f). The unvoiced excitation is modelled by a white noise generator with spectrum N (f). To mix voiced and unvoiced excitation, one can adjust the signal amplitude of the impulse generator (v) and the noise generator (u). The output of both generators is then added and fed into the box modelling the vocal tract and performing the spectral shaping with the transmission function H (f). The emission characteristics of the lips is modelled by R (f). Hence, the spectrum S (f) of the speech signal is given as:

$$(f) = \big(vP(f) + uN(f)\big)H(f)R(f) = X(f)H(f)R(f) \qquad (2.1)$$

2.3 Automatic Speech Recognition Systems

The underlying assumption behind any speech recognition system is that the waveform of a speech signal that comes out of a speaker's vocal apparatus is a realization of the concept that was in the form of symbols in his/her mind. When a source conceives an idea to speak out, it was understood symbolically. The moment it gets out to the channel, it materializes in the form of speech signals or sound waves. Thus, one direct and possible approach for a computer based Speech Recognition System to recognize an utterance is inferring the original symbols from the speech signals [5].

The next step is to design and implement components the human speech perception and production system. The development of an ASR system is complex and requires careful

analysis, design and implementation of its individual parts. Figure 2.3 represents the basic structure of a Speech Recognition System.

The signal processing front-end of a speech recognizer is responsible for the extraction of the features from the speech waveform. The acoustic signal is converted into a parametric representation which is generally at a lower information rate than the original signal. Further processing and pattern matching schemes can then be applied to the encoded representation. Most feature extraction methods involve an analysis of the short term spectral characteristics of the waveform. A commonly used set of features for representing the local spectral properties of the signal is the cepstral coefficients and their temporal derivatives.

The actual recognition stage makes use of an Acoustic Model and a Language Model to arrive at a hypothesis for the spoken sentence. If the specific application allows adaptation, the output can be used to adjust either one or both models or even the signal processing stage. The output of the second stage recognition pass using adapted models can show considerably lower error rates. Both acoustic and Language Models may consist of multiple parts. The vocabulary and the Language Model represent the syntactic and semantic properties of the speech to be recognized whereas the Acoustic Model is responsible for mapping of the feature stream to individual words. In large vocabulary speech recognition the Acoustic Models make use of a pronunciation dictionary which translates words into smaller units reflecting the pronunciation of each word in the vocabulary. However, the pronunciation information may also be represented by a model that translates words or word sequences into pronunciation networks [19].

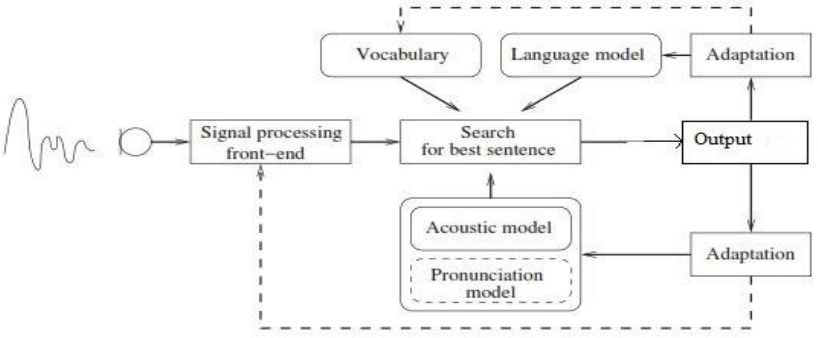

Figure 2. 3*: *Basic structure of a Speech Recognition System [18]

The final role of the recognizer is to affect a mapping between sequences of speech vectors and the required underlying symbol sequences. Modeling techniques and matching algorithms will be used to find out the required association between hypothesized sequences of words and the uttered speech. This module is referred to as Acoustic Modeling. One important approach of Acoustic Modeling, the Hidden Markov Modeling. It is based on the probabilistic approach that the formation makes its decision about which words were spoken.

2.4 Automatic Speech Recognition Approaches

Following design and implementation of human speech perception and production system comes developing Speech Recognition Systems. Generally there are three basic approaches for developing Speech Recognition Systems [18].

- Acoustic-phonetic
- Pattern-recognition
- Artificial Intelligence.

2.4.1 Acoustic Phonetic

The acoustic phonetic-approach is based on the theory of acoustic phonetics that postulates that there exists finite, distinctive phonetic units in spoken language and that the phonetic units are broadly characterized by a set of properties that are manifested in the speech signal or its spectrum over time. Even though the acoustic properties of phonetic units, are highly variable both with speakers and with neighboring phonetics units, it assumes that the rules governing the variability are straight forward and can readily be learned and applied in practical situations. The problem with this approach is to decode phonemes lattice to it a word string called lexical access problem.

2.4.2 Pattern-Matching Approach

The pattern-matching approach involves two essential steps namely, pattern training and pattern comparison. The essential feature of this approach is that it uses a well formulated mathematical framework and establishes consistent speech pattern representations, for reliable pattern comparison, from a set of labeled training samples via a formal training algorithm. A speech pattern representation can be in the form of a speech template or a statistical model (e.g. HMM) and can be applied to a sound (smaller than a word), a word, or a phrase. In the pattern comparison stage of the approach, a direct comparison is made between the unknown speeches (the speech to be recognized) with each possible pattern learned in the training stage

in order to determine the identity of the unknown according to the goodness of match of the patterns. Patter matching approach has two methods namely template approach and stochastic approach.

Template Machining: - Template based approach has a collection of prototypical speech patterns. These patterns are stored as reference patterns representing the dictionary of words. Speech is recognized by matching an unknown spoken utterance with each of these reference templates and selecting the category of the best matching pattern. Normally Templates for entire words are constructed. Errors due to segmentation or classification of smaller acoustically more variable units such as phonemes can be avoided. This approach is simple. It is the process of matching unknown speech against a set of pre- recorded words or template in order to find the best match. This approach has the benefit of using perfectly accurate word models; but this has drawback that the pre-recorded templates are fixed. So variations in speech signals can only be modeled by using many templates per word, which certainly becomes impractical. Template training and matching become prohibitively expensive or impractical as vocabulary size increases beyond a few hundred words. This method is rather inefficient in terms of both required storage and processing power needed to perform the matching. Template matching is also tediously speaker dependent. Continuous speech recognition is not possible using this approach [18].

Stochastic (Statistical) Approach: - Stochastic modeling [18] entails the use of probabilistic models to deal with uncertain or incomplete information. In speech recognition, uncertainty and incompleteness arise from many sources; for example, confusable sounds, speaker variability, contextual effects, and homophones words. Thus, stochastic models are particularly suitable approach to speech recognition. The most popular stochastic approach today is hidden Markov modeling. A hidden Markov model is characterized by a finite state Markov model and a set of output distributions. The transition parameters in the Markov chain models, temporal variability, while the parameters in the output distribution model, spectral variability. These two types of variability are the essence of speech recognition. Compared to template based approach, hidden Markov modeling is more general mathematical foundation.

A template based model is simply a continuous density HMM, with identity co- variance matrices and a slope constrained topology. Although templates can be trained on fewer instances, they lack the probabilistic formulation of full HMMs and typically under performs HMMs. Compared to knowledge based approaches; HMMs enable easy integration of

knowledge sources into a compiled architecture. The side effect of this is that HMMs do not provide much insight on the recognition process. As a result, it is often difficult to analyze the errors of an HMM system in an attempt to improve its performance. Nevertheless, prudent incorporation of knowledge has significantly improved HMM based systems.

2.4.3 Artificial Intelligence

The main idea of the AI approach is to collect and employ knowledge from a number of sources in order to solve a problem in question. The knowledge sources are wide ranging from the fields of acoustic, lexical, syntactic, semantic and pragmatic knowledge [5] Most important techniques in the AI approach are the use of an expert system for segmentation and labeling of the acoustic signal; learning and adaptation over time; and the use of artificial neural networks (ANNs) or distinguishing between similar sound classes and learning the relations between all known inputs and phonetic data. The neural networks can represent a separate structural approach to speech recognition or can be regarded as an implementation architecture possibly incorporated in any of the three classical speech recognition approaches [5].

Researchers developed recognition system by using acoustic phonetic knowledge to develop classification rules for speech sounds. While template based methods provided little insight about human speech processing, but these methods have been very effective in the design of a variety of Speech Recognition Systems. On the other hand, linguistic and phonetic literature provided insights about human speech processing. However, this approach had only partial success due to the complicatedness in quantifying expert knowledge. Another problem is the integration of levels of human knowledge i.e. phonetics, lexical access, syntax, semantics and pragmatics.

2.5 Acoustic Model

According to [19], a source channel mathematical model or a type of generative statistical model is often used to formulate speech recognition problems. As illustrated in Figure 2.4 [19] the speakers mind decides the source word sequence W that is delivered through his or her text generator. The source is passed through a noisy communication channel that consists of the speaker's vocal apparatus to produce the speech waveform and the speech signal-processing component of the speech recognizer. Finally, the speech decoder aims to decode the acoustic signal X into a word sequence W, which is in ideal cases close to the original word sequence W. A typical, practical speech-recognition system consists of basic components shown in the dotted box of Figure 2.4.

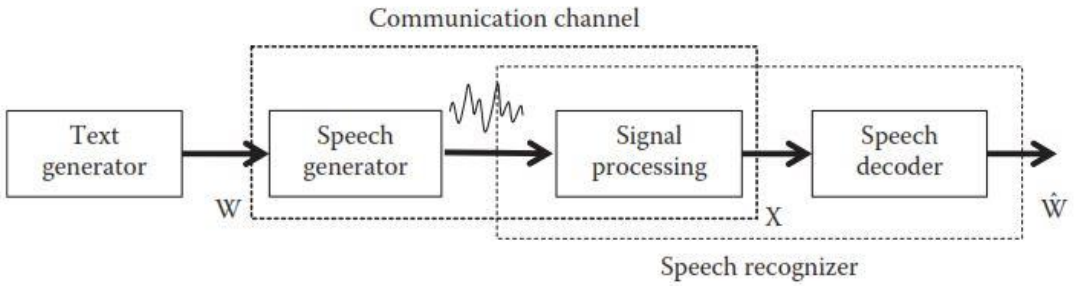

***Figure 2.4*:** Source-channel *model for a typical speech-recognition system [19]*

Applications interface with the decoder to obtain recognition results that may be used to adapt other components in the system. Acoustic Models include the representation of knowledge about acoustics, phonetics, microphone and environment variability, gender and dialect difference among speakers, etc.

Language Models refer to a systems knowledge of what constitutes a possible word, what words are likely to co-occur, and in what sequence. The semantics and functions related to an operation a user may wish to perform may also be necessary for the Language Model. Many uncertainties exist in these areas, associated with speaker characteristics, speech style and rate, the recognition of basic speech segments, possible words, likely words, unknown words, grammatical variation, noise interference and the confidence scoring of results. A successful speech-recognition system must contend with all of these uncertainties. The acoustic uncertainty of the different speaking styles of individual speakers are compounded by the lexical and grammatical complexity and variations of spoken language, which are all represented in the Language Model.

As shown in figure 2.4, the speech signal is processed in the signal-processing module that extracts salient feature vectors for the decoder. The decoder uses both acoustic and Language Models to generate the word sequence that has the maximum posterior probability for the input feature vectors. It can also provide information needed for the adaptation component to modify either the acoustic or Language Models so that improved performance can be obtained.

2.5.1 Signal Processing and Feature Extraction

[20] Indicated that, every other component in a Speech Recognition System depends on two basic subsystems: signal processing and feature extraction. The signal processing sub-system works on the speech signal to reduce the effects of the environment (e.g., clean vs. noisy

speech), the effects of the channel (e.g., cellular/land-line phone versus microphone). The feature extraction sub-system parametrizes the speech waveform so that the relevant information (the information about the speech units) is enhanced and the non-relevant information (age-related effects, speaker information, and so on) is mitigated. There are different methods that can be used for extracting parameters of a speech:-

- production-based
- perception based, by simulating the effect that the speech signal has on the speech perception system
- Signal based, method to describe the signal in terms of its fundamental components.

Regardless of the method employed to extract features from the speech signal, the features are usually extracted from short segments of the speech signal. This approach comes from the fact that most signal processing techniques assume the vocal tract as stationary, but speech is non-stationary due to constant movement of the articulatory during speech production. However, due to the physical limitations on the movement rate, a segment of speech sufficiently short can be considered equivalent to a stationary process. This approach is commonly known as short-time analysis.

Signal based Analysis

The methods in this type of analysis disregard how the speech was produced or perceived. The only assumption is that the signal is stationary. Two methods commonly used are filter banks and wavelet transforms. Filter banks estimate the frequency content of a signal using a bank of band pass filters, whose coverage spans the frequency range of interest in the signal (e.g., 100-3000Hz for telephone speech signals, 100-8000 Hz for broadband signals). The most common technique for implementing a filter bank is the short-time Fourier transform (STFT). It uses a series of harmonically related basis functions to describe a signal.

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

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

Production based analysis

The speech production process can be described by a combination of a source of sound energy modulated by a transfer (filter) function. This theory of the speech production process is usually referred to as the source filter theory of speech production. The transfer function is determined by the shape of the vocal tract, and it can be modeled as a linear filter. However, the transfer function is changing over time to produce different sounds.

The source can be classified into two types. The first one is which is responsible for the production of voiced sounds (e.g., vowels, semivowels, and voiced consonants). This source can be modeled as a train of pulses. The second one is related to unvoiced excitation. In this type, this source can be modeled as a random signal.

Even though this model is a decent approximation of the speech production, it fails on explaining the production of voiced fricatives. Voiced fricatives are produced using a mix of excitation sources: a periodic component and an aspirated component. Such mix of sources is not taken into account by the source-filter model. Several methods take advantage of the described linear model to derive the state of the speech production system by estimating the shape of the filter function. There are three most popular production-based analyses: spectral envelope, linear predictive analysis and cepstral analysis.

Perception-based Analysis

Perception-based analysis uses some aspects and behavior of the human auditory system to represent the speech signal. Given the human capability of decoding speech, the processing performed by the auditory system can tell us the type of information and how it should be extracted to decode the message in the signal. Two methods that have been successfully used in speech recognition from this method of analysis are; Mel-Frequency Cepstrum Coefficients (MFCC) and Perceptual Linear Prediction (PLP).

2.5.2 Mel-Frequency Cepstrum Coefficients (MFCC)

The Mel-Frequency Cepstrum Coefficients is a speech representation that exploits the nonlinear frequency scaling property of the auditory system. This method warps the linear

spectrum into a nonlinear frequency scale, called Mel. The Mel- scale attempts to model the sensitivity of the human ear and it can be approximated by the following formula:

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

2.6 Language Modeling

The idea of word prediction with probabilistic word prediction models called N gram models, which predict the next word from the previous N −1 N-gram models words. Such statistical models of word sequences are also called Language Models (LM). Computing the probability of the next word will turn out to be closely related LMS to computing the probability of a sequence of words [9].

The Language Model together with the Acoustic Model converts the input acoustic signal to a string of words which is called the decoding phase. So, a Speech Recognition System asks: - Given some acoustic observation X, what is the most likely sentence out of all the sentences in the language? Where, X is a sequence of individual observations obtained by segmenting the input wave form into representative chunks of particular durations:

$$X = x1, x2, x3, x4, x5 \ldots\ldots\ldots xn \quad (2.2)$$

Now let us define a sentence W as a string of words w:

$$W = w1, w2, w3, w4, w5 \ldots\ldots\ldots wn \quad (2.3)$$

The division of Acoustic Modeling and Language Modeling can be succinctly de- scribed by the fundamental equation of statistical speech recognition:

$$\hat{W} = \underset{w=L}{\text{argMax}} = \frac{P(w)P(X/w)}{P(X)} \quad (2.4)$$

Where for the given acoustic observation or feature vector sequence in equation 2.2 the goal of speech recognition is to find out the corresponding word sequence equation 2.3 that has the maximum posterior probability P (W/X) as expressed with equation2.4. Since the maximization of equation 2.4 is carried out with the observation X fixed, the above maximization is equivalent of the maximization of the numerator:

$$\hat{W} = \underset{w=L}{\text{argMax}} = P(W)P(x/w) \quad (2.5)$$

2.7 Related Works

A lot of work has been done on Automatic Speech Recognition Systems for different languages. These works ranges from the activities involving development of isolated Speech Recognition Systems to the development of Spontaneous Speech Recognition Systems. The main area which is focused of this is to develop spontaneous ASR system for Tigrigna. Therefore conducting related to the subject area of speech recognition has a vital role on choosing appropriate mechanisms on solving problems and on enhancing the performance of the recognizer.

Different researches have been conducted for the past decade in the area of automatic speech recognition to mention same of reviewed:-Speaker-independent Dictation of Chinese Speech with 32K Vocabulary: they developed speaker-independent, word-based dictation. A deliberately designed 120-speaker database was built for training ; inter-syllable context ,tonal and endpoint dependent Acoustic Model are applied with promising MFCC feature; Two-pass acoustic matching accelerates the recognition making fully advantage of the monosyllabic structure of Chinese speech; A complete word bigram and trigram serve as language processing module. They reported that the system reaches 90% character accuracy performing in almost real-time on Pentium PC [21].

Large Vocabulary Continuous Speech Recognition in Greek: Corpus and an Automatic Dictation System: in this work the researchers presents the creation of the first Greek Speech Corpus and the implementation of a Dictation System for workflow improvement in the field of journalism. They evaluated the word error rate (WER) and the recognition time (xRT) for the three Acoustic Models: Unisex for the gender-independent model they have got 21.01% 6.16 xRT for Male gender dependent model they have got 19.27% WER and 5.78xRT for Female gender Dependent model they have got 20.85 WER and 5.85xRT. As the final step of their work, they developed a graphical user interface (GUI) in order to complete the Greek Dictation System, named Logotypografos1.0. This tool was created using the MFC C++ library for the graphical interface.

Automatic Speech Recognition for Amharic also started in 2001 when [22] developed a speech recognizer for Amharic that recognizes a sub-set of isolated consonant-vowel (CV) syllable using the HTK (Hidden-Markov Modeling Toolkit). Forty one CV syllables of Amharic language out of 234 was selected. Speech data of the selected CV syllables has been recorded from eight people (4 male and 4 female) with the age range of 20 to 33 years. The average

recognition accuracies achieved were 87.68% and 72.75% for speaker dependent and speaker independent systems, respectively. As indicated by the researcher, the result seems to be low compared to systems developed for other languages. This might be due to problems of the recording environment and insufficient training speech data.

As a continuation of [22] [14] had developed sub-word based isolated word recognition systems for Amharic using HTK. The sub-word units used in his experiment are phones, tri-phones, and CV-syllables. He considered 20 phones (out of 39) and 104 CV syllables, which are formed using the selected phones. Speech data of 170 words, which are composed of the selected sub-word units, have been recorded from 20 speakers (speech of 15 speakers for training and the remaining for testing). Speaker dependent phone-based and tri-phone-based systems have an average recognition accuracy of 83.07% and 78% respectively.

Phone-based and tri-phone-based speaker independent systems have an average recognition accuracy of 72% and 68.4% respectively. According to [14] a comparison of the different sub-word units revealed that the use of CV syllables has led to relatively poor performance. This is a contradiction with recent findings of [15] that attributed the poor performance of [14] CV syllable-based recognizers to the use of insufficient speech data. [23] Further, investigated the possibility of developing large vocabulary, continuous speech and speaker independent Amharic speech recognizer [23] had developed both phone-based and tri-phone based recognizers using HTK. For training and testing recognizer, he used the speech data, which amounts to of 8000 sentences read by 80 people, recorded by Solomon et al. (2005). The best recognizer was a tri-phone based recognizer which has 76.2% word recognition accuracy.

Furthermore [9] and [10] in 2015 have emerged as continuation of the development of Amharic speech recognition by different researchers which leads from read speech to Spontaneous Speech Recognition System.

[9] Developed Spontaneous speech recognizer for Amharic, using HTK, with speech data size of 36 speakers with 3 hours and 10 minutes of training data. The test data was prepared with 104 utterances consisting of 820 unique words from 14 speakers, which is 10 of them involved in training and 4 of them did not involve in training. The Language Model used was bigram Language Model developed using HLStats and word networks built from this bigram using HBuild tool.

Another important step towards the development of Amharic speech recognition also Developed and implemented two different dictation application systems by [10].The first one

is dictation application of BRANA (ብራና) in which the entire application is developed with java programming language and it is a PPT (push to talk) prototype which shows a good performance in writing down the recognized results as per the speech recognizer performance. The second one dictation application which is an extension of application part of an Add-On for open office.

According to [10] the system performs 50% word accuracy with small training corpus (90 min) and the performance also increases as the training corpus increases. In this work testing of the application with continuous (read) speech recognizer was also performed. The experimental results show that performance of continuous speech recognizer is better than that of the spontaneous speech recognizer that shows the system to perform better on the recognizer that have a better recognition accuracy.

Research in Automatic Speech Recognition for Tigrigna is not much matured as that of Amharic. The only (as the knowledge of the researcher) available is [7] which is entailed as HMM Based Large Vocabulary, Speaker Independent Continuous Tigrigna Speech Recognition conducted in HTK environment.

The researcher, in his work, used database comprised of 250 utterances that are used for training and 50 sentences for testing and evaluation. The data is pre-processed in line with the requirements of the HTK toolkit. Furthermore the researcher attempted to build speech to text conversion for Tigrigna language using the statistical approach. In order to support the Acoustic Models, a bigram Language Model is constructed. In addition, pronunciation dictionary is prepared and used as an input. In addition to the mono-phone based speech recognizer, the researcher tries a tri-phone based system in which the left and right contexts consider and modelled. According to the researcher, performance tests also conducted at various stages using the training and test data. As a result of this the researcher arrived at 60.20% word level correctness, 58.97% word accuracy, and 20.06% sentence level correctness are obtained.

CHAPTER THREE

3. THE HMMS AND CMUSPHINX

This chapter deals with the well-known modeling technique (HMMs) which is employed in this study this information has been extracted from [24]*; and explains in detail the CMUSphinx toolkit that helps to develop and implement HMM based speech recognizers. The discussion of the toolkit is mainly based on the manual prepared by* [17], [25], *and* [26] *mention will be made if otherwise.*

3.1 Introduction

An HMM-based system, like all other Speech Recognition Systems, functions by first learning the characteristics (or parameters) of a set of sound units, and then using what it has learned about the units to and the most probable sequence of sound units for a given speech signal. The process of learning about the sound units is called training. The process of using the knowledge acquired to deduce the most probable sequence of units in a given signal is called decoding, or simply recognition.

This technology allows the system to get audio input transcribed or used to interact with a system. A Speech Recognition System can handle either a unique speaker or an infinite number of speakers. Modern Speech Recognition System are based on HMMs.

In this chapter problems related to HMM applications are analyzed. Particular attention is put on computing algorithms employed in the ASR. The reason why HMM methods are still preferred to other theories lies in the fact that well tested computation saving algorithms such as the Forward algorithm on subsection 3.3.1, the Viterbi algorithm on subsection 3.3.2 and the Baum-Welsh re- estimation algorithm on subsection 3.3.3 understanding speech recognizers architecture and performance have a critical role on selecting appropriate decoding and training tool. The Sphinx Speech Recognition System is an open source and high performance developed by the CMU Sphinx project that can be used in building speech recognition applications. It includes resources for acoustic and Language Modeling. Hence Sphinx is available as a complete Speech Recognition System for training and decoding. Section 3.4 briefly explains related to the available versions of Sphinx with their predominance and why Sphinx was selected also discussed.

3.2 HMMs

HMM theory is based on doubly embedded stochastic processes. Each model can visit different states aij and generate an output Ot in subsequent time steps. In these systems, a first stochastic process is responsible for the state transitions, while the second process generate the output depending on the actual state. This structure makes HMMs able to model phenomena in which a non-observable part exists. An HMM is specified by the following components:

- $Q = \{q_1, q_2, q_3, \ldots q_N\}$ A set of N reachable states.
- $A = \{a_{01}, a_{02}, a_{03}, a_{04}, \ldots\ldots\ldots\ldots\ldots, a_{nn}\}$ A transition probability matrix A, each aij representing the probability of moving from state i to state j. where
  $\sum_{j=0}^{n} aij = 1 \forall i$
- $B = \{b_j(o_t) = o_1, o_t, o_3 \ldots\ldots\ldots\ldots o_T\}$ is the observation symbol probability distribution
- $\pi = \{\pi_1, \pi_2, \pi_3, \pi_4 \ldots\ldots\ldots \pi_n\}$An initial probability distribution over states, is the probability that the Markov chain will start in state i. Some states j may have πj =0, meaning that they cannot be initial states. Where $\sum_{i=0}^{n} \pi_i = 1 \forall i$thus the probability of state 1 being the first state can be represented either as a_{01} or as π_1.

A first order Hidden Markov model instantiates two simplifying assumptions. The first one is: - First Order Markov Assumption: - As with a first order Markov chain the probability of particularity state is depending only on the previous state

$$P(qi \vee q1 \ldots\ldots q_{t-1}) = P(qi \vee q_{t-1}) \quad (3.1)$$

The second one is: - output independent assumption, the probability of an output observation oi is dependent only on the state that produced the observation qi and no on any other states or any other observations.

$$P(oi \vee q1 \ldots\ldots qi - 1, o1, o2 \ldots\ldots oT) = P(oi \vee q_{t-1}) \quad (3.2)$$

3.3 HMMs and ASRs

When applying HMMs to the speech recognition task three fundamental problems have to be dealt with:

- Problem 1 (**Computing Likelihood**): Given an HMM $\lambda = (A, B)$ and an observation sequence O, determine the likelihood$Prob(O \vee \lambda)$.
- Problem 2 (**Decoding**): Given an observation sequence **O** and an HMM $\lambda = (A, B)$, discover the best hidden state sequence Q.
- Problem 3 (**Learning**): Given an observation sequence O and the set of states in the HMM, learn the HMM parameters A and B.

3.3.1 Computing Likelihood: The Forward Algorithm

The most natural measure of the likelihood of a model λ given the observation sequence O would be $Prob(\lambda \vee O)$.however the available data do not allow us to characterize this statistic during the training process.

A way to overcome this problem is to choose as likelihood measure$Prob(O \vee \lambda)$, the probability that the observation sequence O is produced by the modelλ. A straight forward way to evaluate $Prob(O \vee \lambda)$would be to enumerate all possible state sequences in the time interval $[1, T]$. then $Prob(O \vee \lambda)$would simply be the sum of all probabilities that model λ generates the observation sequence O passing through each state sequence. This method, even if theoretically simple, is computationally impracticable, depending on the huge number of state sequences to deal with in normal applications.

The forward algorithm is a kind of dynamic programming algorithm, i.e., an algorithm that uses a table to store intermediate values as it builds up the probability of the observation sequence. The forward algorithm computes the observation probability by summing over the probabilities of all possible hidden state paths that could generate the observation sequence, but it does so efficiently by implicitly folding each of these paths into a single forward trellis.

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

$$\alpha t(j) = P(o1, o2, o3 \ldots ot, qt \vee \lambda) \quad (3.3)$$

Equation 3.3 will be computed by summing over the extensions of all the paths that lead to the current cell. For a given state qjtime t, the value $\alpha t(j)$is computed as:-

$$\alpha t(j) = \sum_{i=1}^{n} \big(\alpha(i)\big) aijbj(oij) \quad (3.4)$$

The three factors that are multiplied Equation 3.4 15 in extending the previous paths to compute the forward probability at time t are:

- $\alpha t(j)$ The previous forward path probability from the previous time step.
- aij The transition probability from previous state $qi$to current state qj
- $bj(ot)$The state observation likelihood of the observation symbol ot given the current state j

From the given two formal definitions of the forward algorithm; the pseudocode and a statement of the definitional recursion can be defined as follow:-

1. Initialization:-

$$\alpha 1(j) = a0jbj(o01) \quad (3.5)$$

2. Recursion (since states 0 and F are non-emitting):-

$$\alpha t(j) = \sum_{i=1}^{n} \big(\alpha t(i)\big) aijbj(oij) \quad (3.6)$$

3. Termination:-

$$\alpha t(j) = \sum_{i=1}^{n} \big(\alpha t(i)\big) aijbj(oij) \quad (3.7)$$

3.3.2 Decoding: Viterbi algorithm

For any model, such as an HMM, that contains hidden variables, the task of determining which sequence of variables is the underlying source of some sequence of observations is called the decoding task.

Decoding*: Given as input an HMM* $\lambda = (A, B)$ *and a sequence of observations* $O = o1, o2, o3 \ldots . oT$ *find the most probable sequence of states* $O = q1, q2, q3 \ldots . qT$

Instead, the most common decoding algorithms for HMMs is the Viterbi algorithm. Like the forward algorithm, Viterbi is a kind of dynamic programming, and makes uses of a dynamic programming trellis.

The Viterbi trellis are for computing the best hidden state sequence for the observation sequence. The idea is to process the observation sequence left to right, filling out the trellis. Each cell of the Viterbi trellis,$v(j)$ represents the probability that the HMM is in state j after

seeing the first t observations and passing through the most probable state sequence, $q0, q1, \ldots, qt-1$ given the automaton λ. The value of each cell $vt(j)$ is computed by recursively taking the most probable path that could lead to this cell. Formally, each cell expresses the following probability:

$$vt(j) = max \underset{q0,q1\ldots t-1}{N} = P(q0, q1, \ldots, qt-1, o0, o1, \ldots, ot-1, qt = j \vee \lambda) \quad (3.8)$$

Note that the most probable path is represented by taking the maximum of all possible previous state sequences max. Given that it is already computed the $q_0, q_1, \ldots, q_{t-1}, o_0, o_1, \ldots, o_{t-1}, q_t$ Probability of being in every state at time $t - 1$, Viterbi probability was also computed by taking the most probable of the extensions of the paths that lead to the current cell. For a given state $q_j$at time t, the value $v_t(j)$ is computed as:

$$v_j = \max_{t-1} N \;\; V_{t-1}(i) a_{ij} b_j(o_j) \quad (3.9)$$

The three factors that are multiplied Equation 3.9 for extending the previous paths to compute the Viterbi probability at time t are:

- $Vt(i)$ The previous Viterbi path probability from the previous time step.
- a_{ij} The transition probability from previous state $qi$to current state qj
- $b_j(o_t)$ The state observation likelihood of the observation symbol ot given the current state j

Generally Viterbi algorithm is identical to the forward algorithm except that except for two differences

- The Viterbi takes the max over the previous path probabilities where the forward algorithm takes the sum.
- The Viterbi algorithm has one component that the forward algorithm doesn't have: back pointers. This is because while the forward algorithm needs to produce an observation likelihood, the Viterbi algorithm must produce a probability and also the most likely state sequence.

Here:-The best state of sequence by keeping track of the path of hidden states that led to each state, and then at the end tracing back the best path to the beginning (the Viterbi back trace) was computed. Finally, formal definition of the Viterbi recursion can be written as follows:

1. Initialization

$$v1(j) = a0jbj(o01)1 < j < N \quad (3.10)$$

$$bt1(j) = 0 \quad (3.11)$$

2. Recursion (since states 0 and F are non-emitting

$$V_t(j) = \underset{t-1}{max} N v_{t-1}(i) a_{ij} b_j(o_t) \quad 1 < j < N;\ 1 < t < T \tag{3.12}$$

$$b_{t1}(j) = \underset{t-1}{argmax} N v_{t-1}(i) a_{ij} b_j(o_t) \quad 1 < j < N;\ 1 < t < T \tag{3.13}$$

3. Termination:

The best Score $P * v_t qF = \underset{t-1}{max} N v_T(i) * a_{i,F}$ (3.14)

4. The Start of back-trace: $qT * b_{tT}(qF) = \underset{t-1}{argmax} N v_T(i) * a_{i,F}$ (3.15)

3.3.3 Training HMMS: The Forward-Backward Algorithm.

The third problem of HMM is the learning (training) problem in which, given the model and an observation sequence, it attempt to adjust the model parameters to maximize the probability of generating the observation sequence. i.e., A and B matrices. Formally,

Learning: *- Given an observation sequence O and the set of possible states in the HMM, learn the HMM parameters A and B. The input to such a learning algorithm would be an unlabeled sequence of observations O and a vocabulary of potential hidden states Q.[20]*

The aim is to adjust the model parameters $(A, B, \pi)$to maximize the probability of the observation sequence, given the model. In fact, given any finite observation sequence as training data, we can only choose $\lambda = (A, B, \pi)$ such that $Prob(O \vee \lambda)$ is locally maximized. Baum introduced an estimation method based on the Forward-Backward algorithm. This method, known as Baum-Welch Re- estimation algorithm, makes use of an old set of model parameters λ to evaluate quantities that are then used to estimate a new set of model parameter, say $\lambda\hat{}$. Baum proved that two cases are possible after the re-estimation process:

- The initial model λ defines a critical point in likelihood function (null gradient), and in this case $\lambda\hat{}. = \lambda$.
- $Prob(O \vee \lambda\hat{}) > Prob(O \vee \lambda)$ And this gives new model for which the observation sequence is more likely.

The re-estimation theory does not guarantee that in the first case the critical point is also an absolute maximum of the likelihood function. Furthermore, the likelihood function is often a very complex function of many parameters that make up the modelλ, and hence finding a local maximum in the re-estimation process is easier than finding an absolute maximum. The three parameters that need to be re-estimated are:

- πi The Initial state.
- aij The transition probability from previous state qi to current state qj

- $bj(o_t)$ The state observation likelihood of the observation symbol o_t given the current state j

A Markov chain of course has no emission probabilities B. Thus the only probabilities needed to train are the transition probability matrix A. The maximum likelihood estimate of the probability of a particular transition between states i and j is given by counting the number of times the transition was taken, which is called $C(i \rightarrow j)$, and then normalizing by the total count of all times is taken any transition from state i:

Let define a useful probability related to the forward probability, called the back- ward probability. The backward probability β is the probability of seeing the observations from time $t+1$ to the end, given that with given state j at time t (and of course given the automatonλ):

$$\beta t(i) = P(o, o, o \ldots . o \vee qt = i, \lambda) \tag{3.17}$$

It is computed inductively in similar manner to the forward algorithm.

1. Initialization
   $$\beta T(i) = aiF \tag{3.18}$$
2. Recursion
   $$\beta_T(i) = \sum_{j-1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j) \tag{3.19}$$
3. Terminate
   $$P(o|\lambda) = \alpha_T(qF) = \beta_0 = \sum_{j-1}^{N} a_{0j} b_j(o_1) \beta_1(j) \tag{3.20}$$

3.4 Background of Sphinx

History of speech recognizers' development shows that there are two major interrelated goals. The first one is to optimize the accuracy of the recognizer and the second One is to allow a known imperfect speech recognizer to be used in a practical scenario. Usually, this scenario imposes constraints to design of the recognizer such that the recognizer has to run in a limited amount of computation resource (These amount of random access memory and the computation power).

The Sphinx team provides four decoders, these are PocketSphinx (used in live applications), Sphinx-II (used in interactive applications), Sphinx-III (state-of- the-art large vocabulary Speech Recognition System) and Sphinx-IV (state-of-the- art Speech Recognition System written entirely in the Java) which will be discussed in subsection 3.3.2.

CMU Sphinx is a collection of several incarnations of Sphinx, a versatile continuous speech recognition tool-kit from the Sphinx group at Carnegie Mellon University in Pittsburgh. It consists of two major kinds of components: trainers and decoders. The trainers (Sphinxtrain and Simple LM) are used to build acoustic and Language Models. These models are one input used by the various Sphinx decoders to transcribe digital audio. The decoders (Sphinx-II, Sphinx-III, and Sphinx-IV) perform the actual speech recognition. On this work the Sphinx-IV will be used reasons will be discussed on subsection 3.3.3.

Sphinx-IV is a versatile and flexible continuous Speech Recognition System and it is at least as good as the Sphinx-III decoder. Since it is Java-based, has good API support for web services, the object oriented is incorporated in the programming language which makes it easier to deal with, has good Java documentation and is well maintained and updated often. In order to use the Sphinx tools in an optimal way, some software are needed such as Perl to run Sphinxtrain scripts, C compiler to compile Sphinx sources, the Java platform and Apache Ant for using Sphinx-IV.

3.4.1 Versions of Sphinx

Sphinx I: was one of the world's first user independent, high performances ASR (Automatic Speech Recognizer) developed by Dr. Kai-Fu Lee in 1987. Sphinx-I support simple word-pair grammars and generalized tri-phones. It was written in the C programming language and was, as described above, the world's first high performance speaker independent continuous Speech Recognition System. It was based on the viable technology of discrete HMMs, i.e. HMMs that used discrete distributions or simple histograms to model the distributions of the measurements of speech sounds.

In a recognition system, the actual process of recognition is guided by a grammar or Language Model that encapsulates prior knowledge about the structure of the language. Sphinx-I used a simple word-pair grammar, that indicates which word pairs are permitted in the language and which are not.

Sphinx-II: Within five years of its introduction, technology based on semi-continuous HMMs was ready to be used, and much of it had been developed at CMU. Sphinx- II came into existence in 1992, and was again a pioneer system based on the new technology of semi-continuous HMMs. The system was developed by Xuedong Huang at CMU, then a postdoctoral researcher working with Professor Raj Reddy. Like Sphinx-I, it was written in the C programming language.

The essence of semi-continuous HMM based technology was that speech was no longer required to be modeled as a sequence of quantized vectors. State distributions of a semi-continuous HMM were modeled by mixtures of Gaussian densities. Rather than the speech vectors themselves, it was the parameters of the Gaussian densities that were quantized. Sphinx-II used four parallel feature streams, three of which were secondary streams derived from a primary stream of 13-dimensional cepstral vectors computed from the speech signal. All components of the feature streams were permitted to take any real value. For each feature stream, Gaussian density parameters were allowed to take one of 256 values (the number 256 being dictated by the largest number represent-able by an 8-bit number). The actual values of the 256 sets of parameters were themselves learned during training.

In order to achieve real-time speeds, the system was hardwired to use 5-state Bakis topology HMMs for all sound units. Each sound unit was a tri-phone, but unlike Sphinx- I, this system did not use generalized tri-phones. Instead, state distributions of the HMMs for the tri-phones were tied, i.e. states of the HMMs for several tri-phones were constrained to have the same distribution.

Sphinx-II also improved upon its predecessor Sphinx- I in being able to use statistical N-gram Language Models during search, where N could be any number, in principle. An N-gram Language Model represents the probability of any word in the language, given the N-1 prior words in a sentence, and is significantly superior to the word-pair grammar used by Sphinx-I as a representation of the structure of a language.

Sphinx-III: The next milestone in recognition technology was marked by the creation of Sphinx-III. It was developed four years after Sphinx-II was unveiled and incorporated technology that allowed the modeling of speech with continuous density HMMs in a fully continuous vector space. No vector space quantization were required at any level. Naturally, this resulted in better recognition performance as compared to its predecessor, Sphinx-II. However, at the same time due to statistical requirements imposed by this technology, large amounts of data were required to train such HMMs well.

Sphinx-III currently has two different decoders, both written by Mosur Ravis- hankar. Both decoders use statistical N-gram Language Models with Ni=3 during search. They differ in several respects, including their search strategies. One decoder, generally referred to as S3.0, uses the flat search strategy. The second decoder has two variants, usually referred to a S3.2 and S3.3. These differ in their ability to handle streaming input and return partial hypotheses

during decoding. S3.2 does not have these capabilities while S3.3 does. Both variants use a Lex- tree based search strategy.

Sphinx-IV: The years after the development of Sphinx-III saw two major advancements in speech research. First, multimodal speech recognition came into existence, with the development of associated algorithms. Second, it became feasible for Speech Recognition Systems to be deployed pervasively in wide-ranging, even mobile, environments.

In multimodal speech recognition, the information in the speech signal is increased by evidence from other concurrent phenomena, such as gestures, expressions, etc. Although these information streams are related, they differ both in the manner in which they must be modeled and the range of contexts that they capture.

With these considerations, by the beginning of the year 2000, it was clear that Sphinx-III would need to be upgraded. Sphinx-III could only use tri-phone context sound units and required a uniform HMM topology for all sound units. It also required a uniform type of HMM, in the sense that the basic type of statistical density used by each HMM would have to be the same, with the same number of modes, for all sound units, regardless of the amount of training data or the type of feature being used for recognition. Sphinx-III moreover was only partially ready for multimodal speech recognition, although it could use multiple feature streams, it could combine them only at the state level.

3.4.2 Why CMU Sphinx?

Programming Languages: - Sphinx-II and III is in C which is quite portable in general. Sphinx-IV is in Java.

Accuracy and Speed:-Very extensive benchmarking of different Sphinx versions is not available; however, the following rough estimates are present. According to these measurements, the accuracy numbers can be summarized as:

Sphinx-IV >= Sphinx-III >>Sphinx-II

In terms of speed

Sphinx-IV>= Sphinx-III >>Sphinx-I

That means Sphinx-II is still the fastest recognizer we had in our inventory. How- ever its accuracy is also the lowest. Sphinx-III contains best Cs recognizer which can be configured as both fast and pretty accurate. Sphinx-IV is possibly the most all round recognizer and contrary to people's belief, it can actually be pretty fast. Partially, Javas development has come to a stage where it can handle heavy loaded computation.

Interfaces: - Sphinx-IV provides the best interface among all. It can be configured by using a configuration file and command line. This advantage can make web development tasks much easier. Usage of Sphinx-II and Sphinx-III requires much more skill in scripting and in general understanding of the program. At the current stage, they are still regarded as systems for expert users.

Platforms: - Sphinx-II, Sphinx-III and Sphinx-IV are all platform independent. However, the use of Java language in Sphinx-IV may allow higher degree of platform independence. It is also of common interest that whether any version of the decoders could be used in an embedded platform. Sphinx-II is perhaps the best recognizer for embedded platforms due to its smaller size and lesser processing time.

Research: - Reasons to use Sphinx recognizer are Sphinx supports continuous HMMs type which is currently a defacto standard of HMM. In the case of doing individual research. For Acoustic Modeling and fast GMM computation, Sphinx-III is actually pretty good research platform for speech recognition. By itself, it supports SCHMM, CHMM and Sub-vector quantized HMM. They represent there different kinds of modeling technique for speech recognitions. For research in search, Sphinx-IV is a good candidate, because it avoids annoying memory management and focus on implementation of the search algorithm. For speaker adaptation or general Acoustic Model research.

Another aspect of Sphinx is that based on Sphinx is its related resources, and with enough knowledge, allows building a lot of different types of applications. By speech applications, mean things like spontaneous, automatic speech server. More importantly, the most common denominator or all these applications, i.e. the recognizer is open-sourced in CMU Sphinx. It also gives a lot of bundling resource such as dictionary, phone list from CMUs speech archive. This is something very important if to have full-control of Speech Recognition System.

For building HMM-based speech recognition, other reasons why Sphinx-III and Sphinx-IV are used as research tools. First of all Sphinx assumes that GMM computation and the search process are separate. The important consequence is that it allows to carry out two different types of research without conflicting each other's. For example, we can try to device a new observation probability without touching the source code of the search routine. At the same time, we can also build a new search algorithm without thinking of the GMM computation. This gives a better advantage in speech recognition research.

The second thing is in training. Most of the time when modeling research has been performed, the only change that has to be made is the algorithm of estimation. Sphinx-Trains Baum-Welch's algorithm solves this problem in two stages. It first dumps the posterior probabilities

count in a separate file and can be easily readable by libraries of Sphinxtrain. This advantage can greatly shorten research time from weeks to dates.

The third reason is in the code-clarity; later recognizers such as Sphinx-III and Sphinx-IV have put readability of the code as one of the most important design goal. Many researchers, not only they want to use the recognizer as a tool. They also want to change the code to suit their purpose. Sphinx-III, for example, has only 4 layers of code from main () to the GMM computation. These become a good advantage for researchers who like to make changes in a speech recognizer.

The final reason is in speed; all Sphinx's decoders are very fast because the speed of the recognizer is always one important concern. This is usually an important factor why is used in this research.

3.4.3 Acoustic Modeling Toolkit

Sphinxtrain: A good decoder cannot live without a good trainer. Sphinxtrain trainer has an equally interesting development history as in the decoder. At the age when Sphinx-I and Sphinx-II is created and widely used internally in CMU.

Experiments of Acoustic Modeling and Language Modeling are usually done by researchers' code. That means individual researchers hack out a tools using C or perl and full fill the aim of training. The obvious problem is data formats might not be exchangeable and caused a lot of problems in advancing speech recognition research. Two skillful programmers occurred in CMU who solved this problem. First is Dr. Ravi Mosur who is the author of Sphinx-II recognizer and Sphinx-III recognizer(s).

The second one is Dr. Eric Thayer who is the author of Sphinxtrain. At the beginning, the two programmers agreed with the data and model format of the two tools and they start to work on the decoder and trainer separately (Ravi on decoder, Eric on trainer.) At the end, Eric was able to build a set of tools which are essential to the training task. It became the pre-body of Sphinxtrain.

3.4.4 Language Modeling Tool kit

CMU LM Toolkit: The version 1 of CMU LM toolkit is created by Roni Rosen- field. Later Philip Clarkson in Cambridge University improves its caching capability and data structure so that it can handle more data with less memory. It becomes V2 of the program and is open-sourced since 1996.

SRILM: SRILM is a toolkit for building and applying statistical Language Models (LMs), primarily for use in speech recognition, statistical tagging and segmentation, and machine

translation. It has been under development in the SRI Speech Technology and Research Laboratory since 1995.

3.4.5 Decoding Toolkit

Sphinx-IV Framework

The Sphinx-IV framework has been designed with a high degree of flexibility and modularity. Figure 3.1 shows the overall architecture of the system. The Sphinx-IV has three principal modules, the Front End, the Decoder and the Linguist get- ting material of the Knowledge Base.

Front End: - the Front End gets a single or several input signals and computes them so that a sequence of Features (vectors) is created. As in Figure 3.1 the Front End comprises one or more parallel chains of replaceable communicating signal processing modules called Data Processors. Supporting multiple chains permits simultaneous computation of different types of parameters from the same or different input signals. This enables the creation of systems that can simultaneously decode using different parameter types, such as MFCC and PLP, and even parameter types derived from non-speech signals such as video [12].

Linguist: - The Linguist generates the Search Graph by translating any kind of standard language, with the aid of pronunciation information contained in a Lexicon, and with the structural information stored in sets of Acoustic Model.

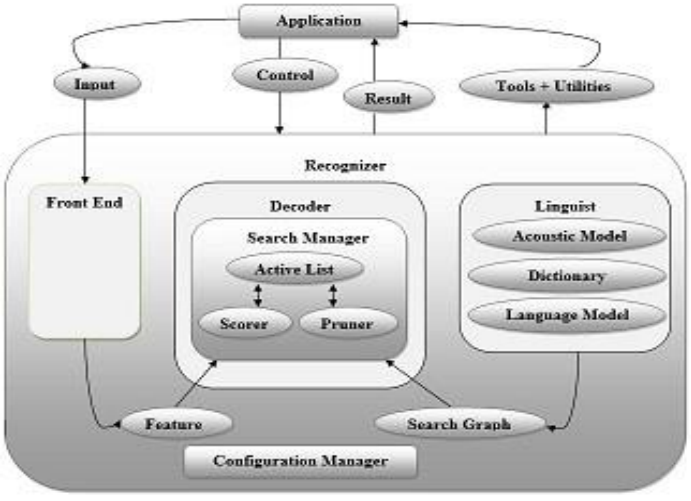

Figure 3.1: *Sphinx-IV Decoder Framework [26]*

A typical Linguist implementation constructs the Search Graph using the language structure as represented by a given Language Model and the topo- logical structure of the Acoustic Model (HMMs for the basic sound units used by the system).

The Linguist may also use a Dictionary (typically a pronunciation lexicon) to map words from the Language Model into sequences of Acoustic Model elements. When generating the Search Graph, the Linguist may also incorporate sub-word units with contexts of arbitrary length, if provided.

Language Model: - The Language Model module of the Linguist provides word-level language structure, which can be represented by any number of plug- gable implementations. These implementations typically fall into one of two categories: graph-driven grammars and stochastic N-Gram models. The graph-driven grammar represents a directed word graph where each node represents a single word and each arc represents the probability of a word transition taking place. The stochastic N-Gram models provide probabilities for words given the observation of the previous n-1 words. The Sphinx-IV Language Model implementations support a variety of formats, including the following:

- Simple Word List Grammar: defines a grammar based upon a list of words. An optional parameter defines whether the grammar loops or not. If the grammar does not loop, then the grammar will be used for isolated word recognition. If the grammar loops, then it will be used to support trivial connected word recognition that is the equivalent of a unigram grammar with equal probabilities.
- JSGF Grammar: supports the Java TM Speech API Grammar Format (JSGF), which defines a BNF-style, platform independent and vendor-independent Unicode representation of grammars.
- LM Grammar: defines a grammar based upon a statistical Language Model. LM Grammar generates one grammar node per word and works well with smaller unigram and bigram grammars of up to approximately 1000 words.
- FST Grammar: supports a finite-state transducer (FST) in the ARPA FST grammar format.
- Simple N-Gram Model: provides support for ASCII N-Gram models in the ARPA format. The Simple N-Gram Model makes no attempt to optimize memory usage, so it works best with small Language Models.

- Large Trigram Model: provides support for true N-Gram models generated by the CMU-Cambridge Statistical Language Modeling Toolkit. The Large Trigram Model optimizes memory storage, allowing it to work with very large files of 100MB or more.

Dictionary: - The Dictionary provides pronunciations for words found in the Language Model. The pronunciations break words into sequences of sub-word units found in the Acoustic Model. The Dictionary interface also supports the classification of words and allows for a single word to be in multiple classes.

Sphinx-IV currently provides implementations of the Dictionary interface to support the CMU Pronouncing Dictionary. The various implementations optimize for usage patterns based on the size of the active vocabulary. For example, one implementation will load the entire vocabulary at system initialization time, whereas another implementation will only obtain pronunciations on demand.

Acoustic Model: - The Acoustic Model module provides a mapping between a unit of speech and an HMM that can be scored against incoming features provided by the Front End. As with other systems, the mapping may also take contextual and word position information into account. For example, in the case of tri-phones, the context represents the single phonemes to the left and right of the given phoneme, and the word position represents whether the tri-phone is at the beginning, middle, or end of a word (or is a word itself). The contextual definition is not fixed by Sphinx-IV, allowing for the definition of Acoustic Models that contain allophones as well as Acoustic Models whose contexts do not need to be adjacent to the unit. Typically, the Linguist breaks each word in the active vocabulary into a sequence of context-dependent sub-word units. The Linguist then passes the units and their contexts to the Acoustic Model, retrieving the HMM graphs associated with those units. It then uses these HMM graphs in conjunction with the Language Model to construct the Search Graph.

Unlike most Speech Recognition Systems, which represent the HMM graphs as a fixed structure in memory, the Sphinx-IV HMM is merely a directed graph of objects. In this graph, each node corresponds to an HMM state and each arc represents the probability of transitioning from one state to another in the HMM.

By representing the HMM as a directed graph of objects instead of a fixed structure, an implementation of the Acoustic Model can easily supply HMMs with different topologies. For example, the Acoustic Model interfaces do not restrict the HMMs in terms of the number of states, the number or transitions out of any state, or the direction of a transition (forward

or backward). Furthermore, Sphinx-IV allows the number of states in an HMM to vary from one unit to another in the same Acoustic Model.

Search Graph: - Even though Linguists may be implemented in very different ways and the topologies of the search spaces generated by these Linguists can vary greatly, the search spaces are all represented as a Search Graph. Illustrated by example in Figure 3.2, the Search Graph is the primary data structure used during the decoding.

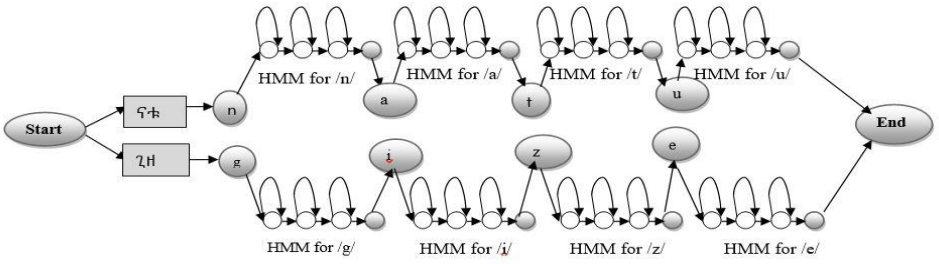

Figure 3.2: *Search Graph data structure used during the decoding process [26]*

Figure 3.2 Example Search Graph. The Search Graph is a directed graph composed of optionally emitting Search States and Search State Arcs with transition probabilities. Each state in the graph can represent components from the Language Model (words in rectangles), Dictionary (sub-word units in dark circles) or Acoustic Model (HMMs). The graph is a directed graph in which each node, called a Search State, represents either an emitting or a non-emitting state. Emitting states can be scored against incoming acoustic features while non-emitting states are generally used to represent higher-level linguistic constructs such as words and phonemes that are not directly scored against the incoming features. The arcs between states represent the possible state transitions, each of which has a probability representing the likelihood of transitioning along the arc. The Search Graph interface is purposely generic to allow for a wide range of implementation choices, relieving the assumptions and hard-wired constraints found in previous recognition systems. In particular, the Linguist places no inherent restrictions on the following:

- Overall search space topology
- Phonetic context size
- Type of grammar (stochastic or rule based)
- N-Gram Language Model depth

CHAPTER FOUR

4. TIGRIGNA PHONETIC AND WRITING SYSTEM

It is useful that the important features of the language, as related to speech recognition, should be properly dealt with. Accordingly, the nature of the basic units of Tigrigna, their contextual behavior and the way they are transcribed are important points in speech recognition system. This chapter is then intended to deal with the Tigrigna sounds and the writing style in relation to the objective of this work in particular and speech recognition in general.

4.1 Introduction

Tigrigna (ትግርኛ) is one of the Ethio-Eritrean Semitic languages, a sub-family of South Semitic languages. The Ethio-Eritrean Semitic languages are further divided into two sub-groups [27]: North Ethiopian and Eritrean Semitic languages, and South Ethiopian Semitic languages. The former comprises Geez, Tigre and Tigrigna. Geez and Tigrigna are common to both Eritrea and Ethiopia, but Tigre is spoken only in Eritrea. South Ethiopian Semitic languages include Amharic as well as over 20 other languages.

The majority of Tigrigna speakers are found in Northern Ethiopia in a region called Tigray where it is the administrative language [27]. According to the census of 2008 conducted by the Central Statistical Authority of Ethiopia, the number of Tigrigna speakers in Ethiopia is about 4.48 million [27]. Tigrigna is also the mother tongue of the Tigrigna ethnic group who mainly inhabit the highlands of Eritrea.

The number of Tigrigna speakers in Eritrea is estimated to be 2.54 million according to the census of 2006 (Lewis 2009) as quoted by [27]. In both countries there are about 6.9 million Tigrigna speakers [27]. Tigrigna serves as the official language of the state of Eritrea along with Arabic.

[27]. There are many regional varieties within the Tigrigna spoken in Eritrea and that spoken in Ethiopia. However, as no dialectal research has been conducted, it is difficult to say how distant the dialects are from each other. The data dealt with in this study mainly reflect the standard variety found in Ethiopia and accent, dialect and other differences among Ethiopian Tigrigna speakers will not be considered.

The spoken word is composed of smaller units of speech, is the Ur-theory that underlies all our modern theories of phonology. This idea of decomposing speech and words into smaller units also underlies the modern algorithms for speech recognition (transcribing acoustic waveforms into strings of text words) and speech synthesis or text to-speech (converting strings of text words into acoustic waveforms) [24].

A Speech Recognition System needs to have a pronunciation for every word it can recognize with in a language [24].This chapter introduces phonetics of Tigrigna language from a computational perspective. Phonetics is the study of linguistic sounds, how they are produced by the articulators of the human vocal tract, how they are realized acoustically, and how this acoustic realization can be digitized and processed. The first section of this chapter will introduce phonetic alphabets for describing these pronunciations. Then following section will introduce the two main areas of phonetics, articulatory phonetics, the study of how speech sounds are produced by articulators in the mouth, and acoustic phonetics, the study of the acoustic analysis of speech sounds based on the book by [28].

4.2 Tigrigna Articulatory Phonetic and consonants

Tigrigna language is primarily comprised of 36 phonemes 7 vowels and 29 consonants phonemes Girmay, cited by [7]. One additional consonant /v/ is inherited and included summing up to a total of 37 phonemes.

Tigrigna sound are produced by the rapid movement of air. Figure 4.1 illustrates parts of sound production in human beings. Most sounds in human spoken languages are produced by expelling air from the lungs through the windpipe (technically the trachea) and then out the mouth or nose. As it passes through the trachea, passes through the larynx, commonly known as the Adams apple or voice box. The larynx contains two small folds of muscle, the vocal folds which can be moved together or apart.

The space between these two folds is called the glottis. If the folds are close together (but not tightly closed), they will vibrate as air passes through them; if they are far apart, they won't vibrate. Sounds made with the vocal folds together and vibrating are called voiced; sounds made without this vocal cord vibration are called unvoiced or voice less. Voiced sounds. The area above the trachea is called the vocal tract, and consists of the oral tract and the nasal tract. After the air leaves the trachea, it can exit the body through the mouth or the nose. Most sounds are made by air passing through the mouth. Sounds made by air passing through the nose are called nasal sounds; nasal sounds use both the oral and nasal tracts as resonating cavities;

According to Girmay, cited by [7] Tigrigna phones are divided into two main classes: consonants and vowels. Both kinds of sounds are formed by the motion of air through the mouth, throat or nose. Consonants are made by restricting or blocking the air flow in some way, and may be voiced or unvoiced. Vowels have less obstruction, are usually voiced, and are generally louder and longer-lasting than consonants.

Because consonants are made by restricting the air flow in some way, consonants can be distinguished by where this restriction is made: the point of maximum restriction is called

the place of articulation of a consonant. Places of articulation, are often used in Automatic Speech Recognition as a useful way of grouping phones together into equivalence classes: [8] and how consonant are voiced called Manner of Articulation.

4.2.1 Place of Articulation

Labial: Consonants whose main restriction is formed by the two lips coming together have a bilabial place of articulation. In Tigrigna these include as shown in table 4.1 [p (ፕ)], [b (ብ)], [pp (ጵ)] and [m (ም)]. The consonants [v (ቭ)] and [f (ፍ)] are made by pressing the bottom lip against the upper row of teeth and letting the air flow through the space in the upper teeth.

Dental: Sounds that are made by placing the tongue against the teeth are dentals. The main dentals in Tigrigna are the [t (ት)], [d (ድ)], [s (ስ)] and [x (ጥ)], which are made by placing the tongue behind the teeth with the tip slightly between the teeth.

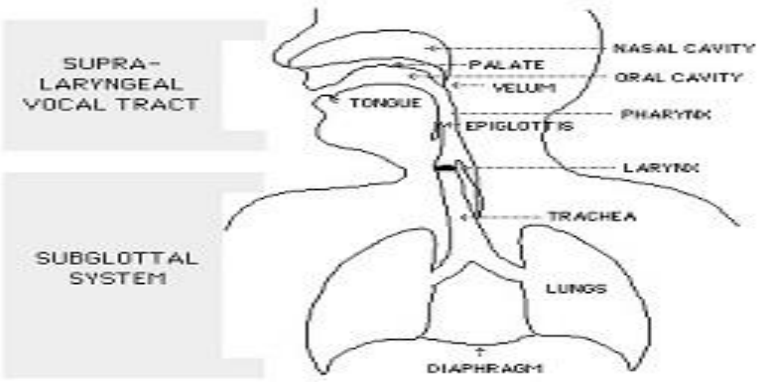

Figure 4.1: Human vocal organs [4]

Alveolar: The alveolar ridge is the portion of the roof of the mouth just behind the upper teeth. Tigrigna speakers make the phones [sh (ሽ)], [zz (ሹ)], [j (ጅ)], and [ch (ች)] by placing the tip of the tongue against the alveolar ridge.

Palatal: The roof of the mouth (the palate) rises sharply from the back of the alveolar ridge. The palatal-alveolar sounds [hh (ሕ)], [kk (ኽ)], [ai (ዕ)] and which are made with the blade of the tongue against this rising back of the alveolar ridge

Velar: The velum or soft palate is a movable muscular flap at the very back of the roof of the mouth.

Glottal: The glottal stop is made by closing the glottis (by bringing the vocal folds together). Below is a summarized discussion of these consonants, with more alphabetical categorization.

4.2.2 Manner of Articulation

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

The Stops: are produce by a complete closure blocking the air commentaries and then releasing it abruptly [25].Stops are also plosives. There are ten stop phonemes out of which three[b(በ),d(ደ),g(ገ)] are voiced and the rest [p(ፐ),t(ተ),k(ከ),p(ጰ), q(ቀ),?(ዐ)] are unvoiced phonemes [p(ፐ),T(ተ),q(ቀ)] are the gluttonized counterparts of [p(ፐ),t(ተ),k(ከ)].

The Fricatives: is made by an incomplete closure (or stricture) which produces friction as the air is forced through its [28] Fricatives are also termed continuants. there are nine fricative phonemes out of which three[z(ዘ),ž(ዠ),’(ኧ)] are voiced and the other six [f(ፈ),S(ሸ),ʾp'(ጰ),š (ሠ),h(ሀ),ḥ (ሐ)] are unvoiced. In addition the velar consonants /k/ and /k'/ are pronounced differently when they appear immediately after a vowel and are not geminated. In these circumstances, /k/ is pronounced as a velar fricative. /k'/ is pronounced as a fricative, or sometimes as an affricate. This fricative or affricate is more often pronounced further back, in the uvular place of articulation, it is represented with [x].

The Affricates: combines a stop with a following continuant but lasts only as long as a single fricative [28].There are three affricate phonemes [j (ጀ), ch (ቸ),’ c (ጨ)] which are voiced, unvoiced and ejective respectively.

The Nasals: is formed when air escapes through the nose. For this to happen, the soft palate is lower to allow air to pass it, whilst a closure is made in the oral cavity to stop air escaping through the mouth [28].there are three nasal phonemes [m(ም)], [n(ን)].

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

Table 4. 1 *Tigrigna Phonemes [28]*

Place of Articulation								
		Labiodental	Bilabial	Dental	Lab.	plain	Glottal	Phary
Stops	Voiced	ብ [b]		ድ [d]		ግ[g]		
	unvoiced	ፕ [p]		ት [t]		ክ[k]	እ[e]	
	ejective	ጰ [p p]		ጥ[x]		ቅ[q]		
Fricatives	Voiced		ቭ[v]	ዝ [z]	ዥ[zz]			ዕ[Aa]
	unvoiced		ፍ[f]	ስ[s]	ሽ[sh]		ህ[h]	ሕ[Hh]
	ejective			ዕ[aa]				
Affricatives	Voiced				ጅ[j]			
	unvoiced				ች[c]			
	ejective				ጭ[cc]			
Nasals	Voiced	ም[m]		ን[n]	ኝ[gn]			
	unvoiced							
	ejective							
Place of Articulation								
		Labiodental	Bilabial	Dental	Lab.	plain	Glottal	uvular
Glides	Voiced				ይ[y]	ው[w]		
	unvoiced							
	ejective							
Uvulars	Voiced							
	unvoiced							
	ejective							
Libio-velars	Voiced				ግ[g] ው[w]			
	unvoiced				ክ[k] ው[w]			
	ejective				ቅ[q] ው[w]			
Libio-Uvulars	Voiced							
	unvoiced							ኽ[K]
	ejective							ቕ[Q]
liquids	edeway			ል[l]				
	nonedeway			ር[r]				

4.3 Tigrigna Vowel Phonemes

Vowels Like consonants, vowels can be characterized by the position of the articulators as they made. The three most relevant parameters for vowels are which is called vowel height, which correlates roughly with the height of the highest part of the tongue, vowel to the front or to the back, which indicates whether this high point is toward the front or back of the oral tract, and the shape of the lips (rounded or not).

Tigrigna has seven vowels. As shown on table 4.2. All are voiced and oral sounds. These vowels determines the shape of the each consonants letters, that is, each letter in Tigrigna is not a single sound rather they are a combination of two sounds, one from vowel and one from consonant [28].

One of the basic differences between consonants and vowels is that in the case of vowels, the air generated by lung will vibrate the vocal cord since the vocal cord is slightly closed when vowels are generated. Moreover, in generating different vowels. The tongue plays an important role.

Table 4. 2 *Tigrigna vowels phoneme* [28]

	Front	Central	Back
High	i(ኢ)	ai (እ)	u(ኡ)
Mid	ie(ኤ)	a (ኣ)	o(ኦ)
Low		a(ኣ)	

The contribution of tongue can be seen in two different aspects [28]. The first one is the way the tongue is positioned in terms of height and the second one is the movement of tongue to the front and back of the mouth. In addition to tongue, the shape of lips has also a contribution in changing the vowels sound when it varies from rounded to non-rounded (flat).finally Vowels differ from consonants in that there is no noticeable obstruction in the vocal tract during their production. Air escapes in a relative unrestricted way through the mouth and/or nose [28]. Vowels can be divided into different categories depending on how they are formulated; Front, Central or Back position of the tongue, wideness /roundness of the constriction position, and place of the tongue (high, mid or low) [28]. Tigrigna the Front Vowel: these are two in number both are not rounded.

The mid front vowel (e): does not occur word finally while the high front (i) occurs medially as well as finally which is discussed on Figure.

The central Vowels: there are three central vowel which are all not rounded .Among these (i) never occurs word finally, whereas the remaining two are found medially and finally.

The Back vowels: these are .Both are rounded and they occur in word medially and word final positions.

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

4.4 The Tigrigna Writing System

The Tigrigna writing system is one variant of what is often referred to as the "Ethiopic" writing system or "Ethiopic syllabary". It is a slight variant of the writing system used for Amharic and for Ge'ez [ግእዝ], the classical language still in use as the liturgical language of Ethiopian and Eritrean Orthodox Christians. The most salient graphical units in this writing system represent a consonant followed by a vowel. Characters representing the same consonant followed by different vowels are similar in shape. For example, here are the characters representing:

/he/, /hu/, /hi/, /ha/, /hie/, /h/ and /ho/:

ሀ ሁ ሂ ሃ ሄ ህ ሆ

As a result, the writing system is usually displayed as a two-dimensional matrix in which the rows contain units beginning with the same consonant and the columns contain units ending in the same vowel as shown in Appendix A. The columns are traditionally known as "orders". The first order, in Tigrinya, represents the vowel /e/, the second the vowel /u/, the third the vowel /i/, the fourth the vowel /a/, the fifth the vowel (really diphthong) /ie/, and the seventh the vowel /o/. The sixth order represents the consonant alone or followed by the vowel /ai/, which, however, is subject to considerable allophonic variation. The literature on Tigrinya suggests that this vowel is always epenthetic, so the sixth order simply represents the consonant by itself. This writing system is sometimes called a "syllabary" because most characters represent a consonant plus a vowel.

In the chart shown in Appendix A., the column Unicode Name gives the Unicode name for the six order in the row. The names for the remaining orders can then be obtained by substituting for ‘a’ the Unicode names for the vowels. These are: e , u, i, a, ie, ai, o. Similarly,

the column Unicode Code gives the Unicode code point, in hexadecimal, for the first order in the row. The column labelled Semi gives the notation commonly found in works by specialists in Semitic language

CHAPTER FIVE

5. EXPERIMENTATION

This chapter describes the design and construction of a Spontaneous speech recognizer for Tigrigna language. First the requirements for the system data preprocessed are stated and the development platform and tools are described. Next an outline of the steps needed to build a speech recognizer is given. Subsequent sections explain these steps in more detail.

5.1 Introduction

In this study an attempt has been made to design and construct spontaneous speech recognizer for Tigrigna language that is capable of recognizing Tigrigna speech. In this attempt, to increase model resolution several experiments have been performed investigating different possibilities of improvement. When building statistical models, the possibility of high performance is always based on a trade of between model complexity and the amount of available data for parameter estimation. A good choice for model structure, based on the comprehension of the sources of variability in the physical phenomena, can help increasing the ability of the model to fit the problem of interest with the minimum number of parameters.

The intention was developing a system that is as general as possible, so that it can be used, with slight modifications and some adaptive training, as a speech interface for many other different applications.

To meet these requirements the recognizer was designed to recognize large vocabulary, spontaneous speech data. This was implemented using phonemes as base unit. Thus, the systems vocabulary was not limited in any way to some fixed set of words. If a new vocabulary is required to be incorporated, what needs to be done is only modifying the Language Model and adding the new word to the pronunciation dictionary. This means that the system can easily be extended and adapted to many applications.

During this study two main directions of improvement have been followed. Context independent (CI) model as a first method. And the second method context dependent (CD) models have been tested with the same data which is a model to add Gaussian terms in the state output probability distribution. In this case the re-estimation procedure is completely responsible for adapting model parameters to different sources of variability in the speech signal.

In order to achieve these goals the work was divided into separate parts. Each targeted to achieve a partial goal. The final product is the combination of all the sub-parts.

It is obvious that the required speech data should be prepared and made available before training the models and building the recognizer. Further, the data should pass through different preprocessing tasks in accordance to the requirements of the various algorithms tools. Which means the first task of any speech recognition development process is data selection and preparation.

The preprocessing part presents the way the data preparation task was performed, the developed Language Model and the prepared dictionary. The training portion deliberates on how the Acoustic Models were developed, the steps followed in training and building the recognizer, and the tools with their accompanying scripts used in the process. It also discusses the mechanisms followed in refining and optimizing the recognizer. The final part is on the testing and performance evaluation of the systems at various stages.

5.2 Data preparation

5.2.1 Design and selection of a Speech Corpus data

As mentioned in the methodology section 1.7.2, there is no data base prepared for Tigrigna language either commercially or prepared by other Tigrigna researcher. In order to fulfill the requirements of large vocabulary spontaneous speech recognizer, spontaneous speech corpus were needed to be developed for training and testing the recognizer. Therefore the goal of this effort was to develop such resources for Tigrigna and then to utilize them to develop a working Speech Recognition System.

The content of the Spontaneous corpus has been designed based on the premise of using a spontaneous speech for spontaneous speech recognition presented in [29]. The spontaneous content serves to ensure phonemic balance and phonetically rich. [30]Here phonetically rich means that the corpus should cover all the phonemes (and hence phones) present in Tigrigna.

Balanced refers to the property that the phonemes should occur in the corpus with the same relative frequency distribution as in naturally spoken Tigrigna.

The collection of spontaneous speech data is considerably more difficult than the collection of read speech data. Tigrigna corpus is developed based on the idea of phonetically rich and balanced Principal. So this served as a good strategy for the corpus development for speech

recognition purposes, since large vocabulary ASR system requires the construction of a phonetically rich and balanced speech corpus for recognition of spontaneous speech.

The first part of this task is then to select data that are going to be used to develop a text corpus with these properties, To meet spontaneous property which is conversational made between two or among more speakers in the form of conversation, discussions of meeting agenda, sports and business agendas without limiting the domain [31]. The spontaneous content was designed to be obtained from DWT, FBC, and FM Mekelle (104.4).

5.2.2 Development of the Speech Corpus

Signal Segmentation and Transcription:-The recorded speech was several hours long and is recorded in different environments which have lots of background noises. In addition to that, it was initially in Mp3 format with 59 speakers. Afterwards, during preprocessing stage, data that do not meet the requirement of spontaneous property were filtered out to four hours and forty five minutes long and twenty four speakers. Then these audio data were converted to .wav format with single channel (mono) which was compatible with CMU Sphinx toolkit.

The data were extremely noisy and were not segmented, so as to meet the requirement for training and testing the data have to be manually segmented into smaller portions using PRAAT, freely distributed speech segmentation and labeling software [19]. It gives full support for the first steps in speech corpus creation. The required steps of signal processing and transcription are described as follows.

Signal Segmentation: - In this step, each utterance were divided into sentences, which were in suitable format for the next processing step. As the transcription is not known, the start- and end-points of the sentences were found manually. As shown in [32] the start-point and end-point of the spontaneous sentences can be very different from fluent read speech. False starts, repairs, and repetitions can appear and the sentence also be very long [33]. Moreover, the speech can be silent at the end of the sentence. Important parts of speech can be hidden in the environmental noise, but it is important to keep this information in the correct segment.

Transcription: - Next to segmentation, the segmented speech data have to be transcribed. Tigrigna language listed in Appendix A for Orthographic annotation were used to transcribe the segmented speech data. Unlike HTK that uses ASCII for transcription, Sphinx supports Unicode language, so no need to transliterate the annotated text into its equivalent ASCII transcription. The segmented speech data have to be transcribed in the form it is exactly

spoken, even in the case of informal language or mathematical expressions, which are typically present in technical speech.

Any special symbols or representations were not used to represent spelling, mispronunciations and foreign words. Punctuation is not noted as it is not important for training a phoneme model-based speech recognizer [32].

Non-speech Event Annotation:- In fact Spontaneous speech differs strongly from read speech is mainly due to the fact that speakers often need to think about succeeding words [32] which makes ill-formed and usually includes redundant information such as disfluencies, fillers, repetitions, repairs and word fragments. In addition, irrelevant information caused by recognition errors is usually inevitably included when spontaneous speech is transcribed. Therefore, an approach in which all words are simply transcribed is not an effective one for spontaneous speech [33]. In order to avoid such problems non speech events have to be modeled. In this work seven non-speech events were modeled as listed in table 5.1.

Table 5. 1: *Non-speech events marks and their description*

<table>
<tr><th colspan="2">Speaker-generated events</th><th colspan="2">Other Speaker Distortion</th></tr>
<tr><th>Representation Mark</th><th>Description</th><th>Representation Mark</th><th>Description</th></tr>
<tr><td>++FP++</td><td>Filled pause</td><td>++OTH++</td><td>Other Speaker</td></tr>
<tr><td>++BR++</td><td>Breath</td><td colspan="2">Background Noise</td></tr>
<tr><td>++LP++</td><td>Lip smack</td><td>Representation Mark</td><td>Description</td></tr>
<tr><td>++HES++</td><td>Hesitations</td><td rowspan="2">++NOISE++</td><td rowspan="2">Stationary Noise</td></tr>
<tr><td>++THC++</td><td>Through clear</td></tr>
</table>

Speaker-generated events: - Those events generated directly by the speaker while the speaker speaking on this work during orthographic transcription five types of such events are used. Here general representations for each non-speech events are used. ++FP++ represents any of filled pauses like ‘ah’ ‘um’, ‘uh’ and the like. ++BR++ represents like ‘inhale’, ‘exhale’ and ++HES++ when the speaker hesitates to generate words.

For examples: <s> ++BR++ ኣብ ናይ ሎማዕንቲ መደብ ስፖርትና ክረምታዊ ምንቅስቃስ ስፖርት ከባብያዊ ምምሕዳር ሰሜን ++HES++ ተዛዚሙ ኣሎ እሱ ክንድህስስ ኢና ++BR++ </s>

<s> ++FP++ ++LP++ ግርም ቅድሚ ሕዚ መፅሓፍቲ ++FP++ ++FP++ ኣይፀንሑናን ስለዝኾኑ እዮም ብፍላይ ኣብዞም ናይ ደርጊ ++FP++ ++FP++ ከምኡ መጣቓሲ ስኢና ንግንፅሎም መፃሓፍቲ ብትግርኛ ዝተፅሓፈ መፅሓፍ ስኢና እናበልና እነማርር ፀኒሕና </s>

Here in those examples, different non-speech events are used in the orthographic transcription directly as it is spoke by the speaker.

Other Speaker Distortion:- ++OTH++ was also used to represent for annotating when another speaker is talking or making sound while the first speaker is talking.

For example: - <s> ብትኽክል ++FP++ ኣርሰናል ድሕሪኡ እናደኸሙ ዝኸድሉ ብሰንደርላንድ ተዓፊኖም ++OTH++ ብፍላይ ብጉልበት ተበለፆም ነይሮም </s> the ++OTH++ used in this example is that another speaker is interfering while the first speaker is talking.

Background Noise: ++NOISE++ is used to represent a background noise since the speech is recorded in a in different environments with different background noises like noises that come from chair and table sounds, murmuring of people around recording area mobile ringing and so on. Example <s> ከንቲባ ቤት ፅሕፈት ++NOISE++ </s>

5.2.3 The Data Set

The trainer learns the parameters of the models of the sound units using a set of sample speech signals. This is called a training database. The database contains information required to extract statistics from the speech the form of the Acoustic Model. Database should have enough speakers recording, variety of recording conditions, enough acoustic variations and all possible linguistic sentences [17] the construction of such database is a major undertaking by itself and involves a number of subtasks. It also demands sufficiently large time and big financial support.

For steps in sub-section 5.2.1 a corpus containing 3524 sentences spoken by twenty four speakers of twelve male and twelve female were prepared as in Table 5.2. The number of female speakers and male speakers are balanced. The utterances are also comprised of all Tigrigna phonemes in many phonetic contexts. Due to this this system may not work for non-native speakers.

A data set must have two parts, training part which is used for training the system and test part used for testing the system's performance Most of the data from the database, consisting of ten female with 1617 sentences and male with 1558 sentences, are used. Usually test part is about 1/10th of the full data size [17]. The remaining two males with a total number of 172 sentences and two females with a total number of 177 sentences were used for test part.

Table 5.2: *Data set used*

Data Set	Gender	Number of Speakers	Number of sentence
Training Data	Female	10	1617
	Male	10	1558
Test Data	Female	2	177
	Male	2	172

5.3 Adaptation of the Speech Corpus to the Speech Recognition System

This step requires processing the speech corpora to conform to the input standards of the Sphinx Speech Recognition System. Besides that, the development a compiler like tool which takes a transcribed corpus as input and produces all the resources as an output required by the Sphinx ASR.

Segmented Audio Data: - The audio (speech) data files are required to be .wav format. These files were used in developing the MFCCs file: set of feature files computed from the audio training data, one for every recording have in the training corpus using which the system will be trained. The audio files used were .wav format, single channel, sampled at 16000 sample per seconded and were renamed as follows.

trF001-001.wav

trF001-002.wav

trF001-003.wav

Transcription file:- Transcription file is a text file listing the transcription for each audio file each line starts with <s> and ends with </s> followed by id in parentheses with parenthesis contains only the file. The following is an example taken from **TigSpeech.transcripion**.

<s> ተወሊደ ዝዓበኹሉ ሸረ ወረዳ መደባ ኢዛና ጣብያ እንዳ ምቸኣል ተኹለ ይበሃል ዓዲ ኦርኪ ፍሉይ ቦታ </s> (trF001-002)

Dictionary preparation: - Usually the first step in building the dictionary is to create a sorted list of the words contained in the transcription file one word per line, with pronunciations (the phonemes that makes up the word).

According to [24] phonemes are the basic units of Speech Recognition System and Speech Recognition System needs to have a pronunciation for every word it can recognize. A phone is a speech sound; that are represented with phonetic symbols that bear some resemblance to

a letter in an alphabetic language. As mentioned in section 4.2 Tigrigna has 36 phonemes 7 vowels and 29 consonants phonemes with one additional consonant /v/ is inherited and summing up to a total of 37 phonemes.

Transcription of the phonemes in the corpus is primarily in a font known as Ethiopic. Those phonemic representations of the font are suitable for CMU Sphinx toolkit. Thus, they need not to be renamed and modified for convenience transcribing. But as per the requirement of CMU Sphinx's pronunciation dictionary phonemes whom makes the word have all acoustic events and words in the transcripts mapped onto the acoustic unit. The phonemic representations of the font were not suitable for the tools to create the dictionary for CMU Sphinx. Due to this they were renamed and modified for convenience in later activities. Appendix A, depicts the phonemes used in their original notation, and notation used in this experiment.

In order to generate the pronunciation dictionary, a system was required and written in order to create the word list out of the transcribed text. For this purpose, a system in figure 5.1 was developed using java programming language. The system creates a list of all the words in the database. Using the prepared sorted word list, it produced a pronunciation dictionary, containing the words and their associated phone level pronunciation in parallel. Finally, a file called **TigSpeech.dic** was created with total of 10731 unique words and was in the following form.

ሀመይ h e m e y

ሀም h e m

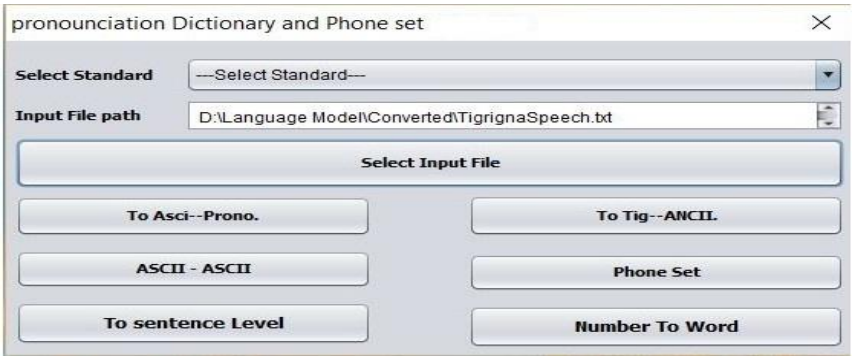

Figure5.2: *Pronoun dictionary and phone list generator*

Phone List generation: - A phone list, which is a list of all acoustic units are used to train models for, The SPHINX does not permit units other than those in dictionaries. In other words, the phone list must have exactly the same units used in dictionaries, no more and no less. Each phone must be listed on a separate line in the file, beginning from the left, with no extra spaces after the phone. Initially the transcription file was segmented in to word level according to their phoneme representation and the system list out unique phoneme from the transcription file, the system in figure 5.1 was used to generate phone list out of previously prepared pronunciation dictionary called **TigSpeech.phone** as follows:

a

ah

ai

Feature vector extraction: - Another important task in data preparation is the extraction of feature vectors i.e. parameterizing the raw speech waveforms into sequences of feature vectors. For training and testing purposes it is reasonable to do this once for all and save the feature vectors in a file called **feat**. Because each utterance is processed a number of times during training it helps to reduce later processing costs. The Sphinx trainer is responsible for translating audio files to feature vector files using Mel scale cepstral coefficients with input parameters listed in table 5.3.

Table 5.3: *Parameters used in feature extraction.*

Parameter	**Value**
Sampling Rate	16Khz
Pre-emphases Coefficient	0.97
Window Size	25ms
Overlap Duration	15ms
Hamming Window	True
Zero Mean	True
Cepstral lifter	22
Number of Cepstral Coefficient	12

The file generated was of the following format:

trF001-001.mfc

trF001-002.mfc

The audio files were encoded to Mel-frequency cepstral coefficient vectors. This choice was based on literature and [17].

5.4 Language Model

After training, it's mandatory to run the decoder to check training results. The Decoder takes a model, tests part of the database and references transcriptions and estimates the quality of the model. During the testing stage, Language Model with the description of the order of words in the language is crucial part of a Speech Recognition System. The Language Model describes the probabilities of the sequences of words in the text and is required for speech recognition there is no Language Model prepared for Tigrigna language yet. In order to evaluate the decoder, Language Model must be prepared first. Therefore, in order to build language model the following steps were followed.

Data collection

The first step of Language Model building is a collection of the data. The amount of data needed depends on the domain and vocabulary. Usually for a good model it needs a significant amount of texts - at least 100mb [20]. This data needs to get text by transcribing existing recordings, collecting data from the web, generating it artificially with scripts and from broadcasting. Generally, the most valuable data is a real-life data. But this task by itself requires sufficiently large time. For this work data were collected form Woyan gazette, Mekalih Tigray gazette and Tigray television an amount of 50MB.

Text cleanup:- Once data is collected it must be cleaned - punctuation removed, sentences split, numbers expanded to text representation. For this purpose the system depicted in figure 5.1 was used splitting paragraphs in to sentence level and it enables representation of numbers to their expanded text.

Language Model Preparation:-The Unigram, Bigram or Trigram Language Models can be generated using either the CMU Language Modeling CMU-LM toolkit or SRILM toolkit which works in Linux environment. During language model preparation the SRILM language modelling tool was used.

SRILM: SRILM is a toolkit for building and applying statistical Language Models (LMs), primarily for use in speech recognition. Training with SRILM is easy, and recommend by [17]. Moreover, SRILM is the most advanced toolkit and up to date. The SRILM toolkit also useful as it allows more options for generating the Language Models such as different

smoothing techniques like Laplace smoothing, Additive smoothing and linear interpolation. For preparing the Language Model 10MB normalized text with 30937 sentence was used. In order to optimize the language model the Laplace smoothing technique was used during the development.

After generating, the Language Model has to be converted to binary file using a converter. Sphinx4 requires DMP format. DMP format can produce file with **Sphinxss_lm_convert** command from Sphinx base. Finally data required for training and testing are organized as follow:-

TigSpeech.dic - Phonetic dictionary

TigSpeech.phone - Phoneset file

TigSpeech.lm.DMP - Language Model

TigSpeech.filler - List of fillers

TigSpeech_train.fileids - List of files for training

TigSpeech_train.transcription - Transcription for training

TigSpeech_test.fileids - List of files for testing

TigSpeech_test.transcription - Transcription for testing

5.5 Training Speech Recognition System

In section 5.4, all the required resources for training and testing the recognizer as well as developing the models were prepared. Here the tasks performed in developing the recognizer and the techniques followed in refining and improving performances is explained.

On this work, as mentioned Sphinx as the main component for designing and decoding the model and Ubuntu 16.04 64bit operating system installed on Intel(R) Core(TM) i5 CPU of 2.20 GHz, 8 GB memory, and 1TB hard disk was used for training, testing and creating the model.

Setting up configuration files:-Before starting the train parameters needed to edit the configuration files in “etc” folder, there are many variables that need to change in the file etc/Sphinx_train.cfg the file’s full configuration will be depicted in Appendix C.

Setup the format of database audio:-As stated the audio files used for training the model are in .WAV format before starting train this needs to be configured as follow

```
$CFG_WAVFILES_DIR = "$CFG_BASE_DIR/wav";

$CFG_WAVFILE_EXTENSION = 'wav';

$CFG_WAVFILE_TYPE = 'mswav';
```

Configure model type and model parameters:-CMU Sphinx supports different types of Acoustic Models; continuous, semi- continuous and phonetically tied. In this work, the training was performed on the continuous (.cont.) HMM type with Context-Independent (CI) and context-dependent (CD) phones model. The Acoustic Model is developed with no skip transition topology and 8 Gaussian mixtures as follows.

```
$CFG_HMM_TYPE = '.cont.' #

#$CFG_HMM_TYPE = '.semi.' #

#$CFG_HMM_TYPE = '.ptm.' #
```

Configure sound feature parameters:-Sphinx uses a rate of 16 thousand samples per second (16 KHz) with 16 bit sample size then this has to be configured in alignment with the audio format and sample rate

```
# Feature extraction parameters

$CFG_WAVFILE_SRATE = 16000.0;
```

5.6 Testing and Evaluation

Testing and evaluation were the final tasks in the process of developing a speech recognizer. After completing training it's critical to test the quality of the trained database in order to select best parameters, understand how application performs and optimize performance. To do that, a test decoding step is needed. The decoding is a last stage of the training process.

Usually, a set of testing data is first recorded, then the sentence error rate (SER) or word error rate (WER) are then found as stated in methodology. The performance of speech recognition systems is usually specified in terms of accuracy [17]. Accuracy can be measured in terms of performance accuracy which is usually rated with word error rate (WER).

WER is a common metric of the performance of a speech recognition or machine translation system. The general difficulty of measuring performance lies in the fact that the recognized word sequence can have a different length from the reference word sequence. The WER is working at the word level instead of the phoneme level. This problem is solved by first aligning the recognized word sequence with the reference (spoken) word sequence using dynamic string alignment. Word error rate can then be computed as:-

$$A = \frac{H-S-I-D}{H} \times 100 \quad (5.1)$$

- H is the total number of words in the reference,
- S is the number of substitution errors,
- I is the number of insertions errors and
- D is the number of deletions errors.

5.7 Results with modelling non-speech

The training was performed in two phases based on non-speech events with CI and CD and the same acoustic and Language Model containing four unseen speaker (speaker whom are not participated in the training) with a total of 349 sentences were used. Table 5.4 shows speakers used during testing with their gender, number of sentences and amount of modelled non speech events. On his step non speech events (such as speaker generated, other speaker generated and back ground noise) were modelled. In training transcription 1285 Filled pause (++FP++), 19 Breaths (++BR++), 228 Hesitations (++HES++), 10 Lip smacks (++LP++) and one through clear (++THC++) with respect to speaker generated events. And also 70 other speakers (++OTH++) and 26 back ground noises were modelled. In testing transcription 170 Filled pause (++FP++), one Breaths (++BR++), 12 Hesitations (++HES++), one Lip smacks (++LP++) and zero through clear (++THC++) with respect to speaker generated events. And one other speaker (++OTH++) and two back ground noise were also modelled.

***Table 5.4*:** Test speakers' information

Speaker Code	Gender	Number of Sentences	Number of non-speech events
spF21	Female	96	22
spF23	Female	81	19
spM004	Male	69	75
spM007	Male	103	80

First testing was performed on the CI model with the CD model set to ‘no ‘ with using 5-state phone models and have 349 utterances spoken by 4 people. The results obtained at these levels are compared and summarized by the following table 5.5.

***Table 5.5*:** Results of CI Model

Speaker Code	% Correct	% Accuracy	% SER
spF21	20.02%	19.577	86.44
spF23	25.04%	24.00%	100.00%
spM004	28.57%	27.57%	86.46
spM007	29.17%	28.17%	94.00
Total	23.08%	23..00%	87.42%

The table displays the accuracy achieved during the CI model testing. And the overall performance of the system with CI model is 23.00%. Whereas the sentence error rate is higher by 6.17% than the word error rate.

Second testing was performed on the CD model by tuning the Gaussian mixture set to single (1), two (2) four (4) and eight (8) Gaussian mixture. The results obtained at those levels are compared and summarized on following tables.

***Table 5.6* :** Results of CD Model with single Gaussian Mixture

Speaker Code	% Correct	% Accuracy	% SER
spF21	29.79%	28.76%	87.50%
spF23	28.76%	26.36%	100.00%
spM004	35.27%	30.27%	86.77%
spM007	31.55%	29.63%	100.00%
Total	30.57%	29.55%	86.98%

Table 5.6 shows that recognition result of single (1) Gaussian mixture the overall recognition accuracy is 29.55%. This indicates that performance improvement is achieved from the CI Model to single Gaussian mixtures. The performances at the word level increase as the mixture is incremented, though the sentence level error rate is almost the same.

***Table 5.7*:** Results of CD Model with two Gaussian Mixture

Speaker Code	% Correct	% Accuracy	% SER
spF21	30.70%	29.26%	87.68%
spF23	29.26%	26.98%	100.00%
spM004	35.99%	30.68%	85.49%
spM007	30.66%	27.54%	100.00%
Total	31.12%	30.12%	86.41%

Table 5.7 shows that recognition result of two (2) Gaussian mixture. The overall recognition accuracy is 30.12%. This indicates that performance improvement is also achieved from single Gaussian mixtures Model to two (2) Gaussian mixtures. The performances at the word level also increase as the mixture is incremented, though the change is very small. In two (2) Gaussian mixtures the sentence level error rate is almost the same.

***Table 5.8*:** Results of CD Model with four Gaussian mixture.

Speaker Code	% Correct	% Accuracy	% SER
spF21	32.09%	31.09%	89.50%
spF23	26.30%	25.30%	100.00%
spM004	36.36%	36.34%	86.5%
spM007	35.27%	35.227%	100.00%
Total	33.33%	33.33%	86.67%

Table 5.8 shows that recognition result of four (4) Gaussian mixture. The overall recognition accuracy is 33.33%. This indicates that performance improvement is also achieved from two (2) Gaussian mixtures Model to four (4) Gaussian mixtures with significant amount of 3.21%. This increment, especially on the performances at the word level for all speakers as the mixture is incremented. Whereas the overall sentence level error rate almost the same with 0.26% decrement from the two (2) Gaussian mixtures Model.

***Table 5.9*:** Results of CD Model with eight Gaussian mixture

Speaker Code	% Correct	% Accuracy	% SER
spF21	32.97%	30.13%	85.50%
spF23	27.36%	37.01%	100.00%
spM004	37.09%	36.36%	83.51%
spM007	35.97%	35.37%	100.00%
Total	36.83%	36.83%	86.37%

Table 5.9 shows recognition result of CD Acoustic Model with eight (8) Gaussian Mixture and with overall recognition accuracy 36.83%. On this stage highest performance improvement is achieved from all the Gaussian mixtures Model. In this model the word level accuracy is increased. As shown from the results the eight (8) Gaussian Mixture performance is accurate than the four (4) Gaussian Mixture by 3.50% and 6.71% from two (2) Gaussian Mixture and finally 7.28% from the single Gaussian Mixture. Form the results obtained, As Gaussian mixture increases the overall recognition accuracy increase. But the sentence level accuracy is very low in all Gaussian mixtures.

Table 5.10 : Summarized Results of all Gaussian mixture CD Model

Gaussian mixture	% Correct	% Accuracy	% SER
Single	30.57%	29.55%	86.98%
Two (2)	31.12%	30.12%	86.41%
Four (4)	33.33%	33.33%	86.67%
Eight (8)	36.83%	36.83%	86.37%

5.8 Result without modelling non-speech

As shown in table 5.10 the performance of the Gaussian mixture with Eight (8) Gaussian mixture model was the highest performance with modelling the non-speech events. In this experiment a training was performed without modelling the non-speech events (such as speaker generated, other speaker generated and back ground noise). Sample sentences taken from main training transcription are shown below as an example:

<s> ኣብ ናይ ሎማዕንቲ መደብ ስፖርትና ክረምታዊ ምንቅስቃስ ስፖርት ከባብያዊ ምምሕዳር ሰሜን ተዛዚሙ ኣሎ እሱ ክንድህስስ ኢና </s>

<s> ግርም ቅድሚ ሕዚ መፅሓፍቲ ኣይፀንሑናን ስለዝኾኑ እዮም ብፍላይ ኣብዞም ናይ ደርጊ ከምኡ መጣቓሲ ስኢና ንግንፅሎም መፃሓፍቲ ብትግርኛ ዝተፅሓፈ መፅሓፍ ስኢና እናበልና እነማርር ፀኒሕና </s>

As can be seen all the non-speech events were filtered out. on this experiment training was performed only based on the HMM type with continuous ('.cont.) and CD with the same acoustic and Language Model containing four unseen speaker (speaker whom are not participated in the training) as shown in experiment one. On this experiment to evaluate the effect of modelling non-speech events only Eight (8) Gaussian mixture model training was performed.

***Table 5.11*:** Results of CD Model with eight Gaussian mixture without modelling non-speech events

Speaker Code	% Correct	% Accuracy	% SER
spF21	22.54%	22.01%	100.00%
spF23	20.78%	20.38%	100.00%
spM004	21.37%	20.93%	94.77%
spM007	19.21%	19.21%	100.00%
Total	21.33%	20.42%	99.6%

Table 5.11 shows Recognition results of CD Model with eight (8) Gaussian mixture without modelling non-speech events the overall recognition accuracy 20.42%. This indicates the system performs 22.01% for spF21 who have a total 96 sentences with 22 non-speech events and the performance of the system in the second speaker who have a total of 81 sentences with 19 non-speech event was 20.38% on the other hand the performance of the system on speaker spM007 who have large amount of non-speech events scores list performance. The performance obtain with speaker spF004 who has large amount of non-speech event with respect to spF21 and spF24 also scores low performance than spF21 who has 22 non-speech events.

5.9 Architecture of Developed Speech Recognizer

Figure 5.2 shows the components of developed speech recognizer as it processes a single utterance taken form the data set indicating the computation of the prior and likelihood. In the figure shows the recognition process in three stages. In the feature extraction or signal processing stage, the acoustic waveform is sampled into frames which are transformed into spectral features. Each time window is thus represented by a vector of around 39 features representing this spectral information as shown in table 5.3

In the acoustic modeling or phone recognition stage, the system compute the likelihood of the observed spectral feature vectors given linguistic units the output of this stage is as a sequence of probability vectors, one for each time frame, each vector at each time frame containing the likelihoods that each phone unit generated the acoustic feature vector observation at that time.

Finally, in the decoding phase, the acoustic model (AM), which consists of the sequence of acoustic likelihoods, plus the dictionary of word pronunciations (Tigrigna Dictionary), combined with the language model (LM) (Trigram Tigrigna Language Model), and output the most likely sequence of words.

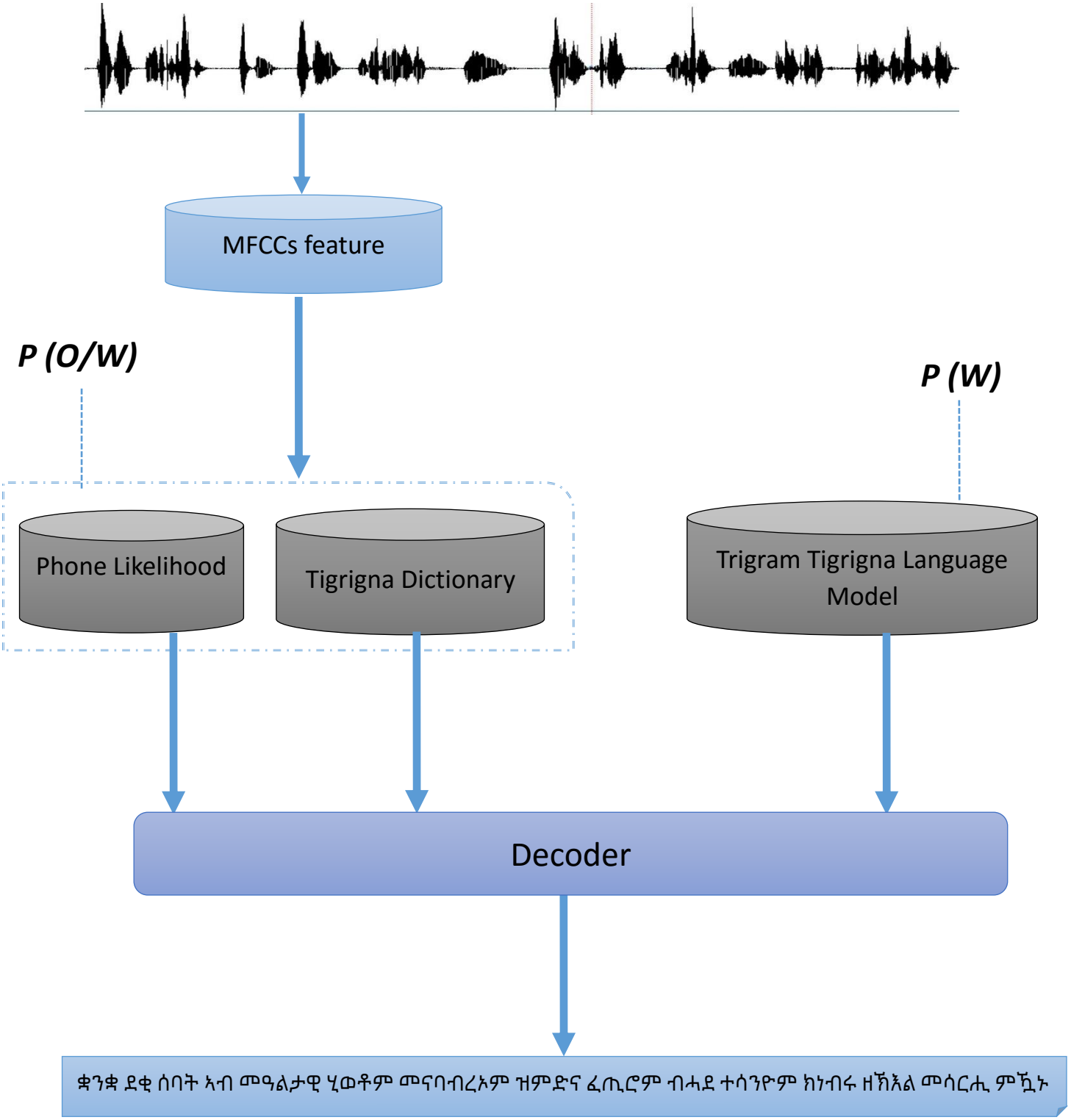

Figure 5.6: A*rchitecture of developed speech recognizer decoding a single sentence.*

5.10 Comparison of Results and Discussion

The Tigrigna spontaneous speech recognizer developed has been tested using test data consisting of 349 utterances with 2999 unique words. The language model used were word probability was trigram language model. As mentioned previously to evaluate the performance of the recognizer the Gaussian mixture was incremented from single Gaussian mixture to eight Gaussian mixture. Therefore after achieving the maximum performance with modelling the non- speech events at eight Gaussian mixture the performance of the system was also evaluated at eight Gaussian mixture without modelling the non-speech event. The results were given in tables 5.10 and 5.11.

During the development of recognizer context Independent mono-phones and context dependent tri-phone with modelling and without modelling non speech events were used. It is also stated by [7], [9] and [10] similarly recognizer developed using context dependent has better performance than the context independent. In this experiment the context dependent's performance has better performance. This is because the movement of the articulators (tongue, lips, and velum) during speech production is continuous and is subject to physical constraints. Thus an articulator may start moving during one phone to get into place in time for the next phone and Spontaneous speech is subjected to such variations. So Context-dependent phones capture an important source of variation by representing a phone in a particular left and right context

Generally the CD Model performance is better than the CI Model by an amount from the least performing of Gaussian mixture (from single Gaussian mixture) 6.55% and from the highest performing of Gaussian mixture (eight (8) Gaussian mixture) by 13.83%. Table 5.10 depicts the summarized results of all the Gaussian mixture Models. The performance of the Gaussian mixture increases from single Gaussian mixture model to multiple Gaussian mixture.

The result clearly indicates that modelling non-speech events positively affects the performance of the system. The word level accuracy decreases from 36.83% in table 5.10 to 20.42 % from table 5.11 and the sentence level accuracy also declined form 13.73% to 0.40%. The performance of speaker spF007 was the lowest performance achieved yet as shown in table 5.4 spM7 has more non speech event and has low performance as well as spM4 had higher performance during

modelled non speech event but in this experiment it declined significantly to 20.93%. As shown from table 5.10, the word recognition accuracy result for the recognizer developed using data set which modelled non-speech events is 36.83% and the analysis result for recognizer which was developed without modelling non-speech events is 20.42%.The recognition accuracy which is found from the modelled disfluencies (non-speech events) is better. As [9] and [34] stated that the spontaneous ASR performance declined due to the occurrence of dis-fluencies with spontaneous speech, similarly the recognition accuracy increases with the modelling of dis-fluencies.

Here the performance results clearly indicates major decline in comparison to the performance obtained in the last researches conducted by [7]. These results were to be expected, since the data set recorded was in a different and noisy environment as stated in section 5.2.2. The nature of the speaker was also another factor for declining the performance since the data used were spontaneous speech data whereas the data used by [7] were read speech.

Even though the performance results show decline in comparison to the performance obtained in the last researches conducted by [7], in this research as indicated large data (spontaneous speech data of 4 hours and 45 minutes for training and 30 minute for testing spoken by 24 persons) were used which better than data used by [7] (read speech data of 250 sentences for training and 50 sentences for testing speaker independent spoken by 12 persons) this shows that an attempt was made to improve the data size and the number of speaker as well.

For Amharic Spontaneous speech recognition developed by [9] with a data size of 3 hours and 10 minutes of data spoken by 36 persons with total 2007 number sentences and 9460 unique words spoken by 36 were used which is better in the number speakers used than in this work whereas in total hours, number of sentences and unique word used in this work were better than data set used by [9].

Depending on the language model as indicated in this work a total of 30936 sentences were used compared to a language model used by [10] in this work larger number of sentences were used but with respect to Corpus of Spontaneous Japanese (CSJ) training data of 510 hours long and 6.84 M words for language modeling, the data used in this work is very small.

CHAPTER SIX

6. CONCLUSION AND RECOMMENDATION

This chapter presents the conclusions drawn from the findings of the experiment and the researcher's recommendations on further actions that can be taken, future research areas and contribution of the work.

6.1 CONCLUSION

On this study, possibility of the speaker independent spontaneous speech recognizer for Tigrigna has been investigated. And different related literatures on Spontaneous Speech generally on ASR for Tigrigna, Amharic, Afaan Oromo and other languages have been reviewed and in order to understand the core principals behind HMM related to speech recognition system as well as manuals and web address of CMU Sphinx also the main part of the review. For this study, among all the approaches of ASR systems the Statistical (stochastic) approach was selected. HMM has been implemented for modeling with the CMU Sphinx for tool and different tools which are compatible with the approach and modeling technique were implemented accordingly.

Spontaneous Speech Recognizer for Tigrigna, has been developed. It has 4 hours and 45 minutes of training data collected from 24 speakers and 30 minutes of test speech collected from 4 unseen speakers were used. The test data was prepared 349 utterances consisting of 2999 unique words. The Language Model used was tri-gram developed using SRILM and Sphinx was also used to convert the Language Model to .DML format which is suitable for Sphinxtrain.

Unlike read speech Spontaneous speech is full of non-speech events (dis-fluencies) those issues have direct influence in the performance of the recognizer as shown in chapter five. On this work non-speech events was properly handled and modelled in experiment one and then in order to evaluate the effect of non-speech events on the performance of the system a second experiment was performed without modelling the non-speech events .

During the training, standard procedure used by Sphinxtrain trainer was followed in order to form the Acoustic Models which is one component of the speech recognizer. And to develop the pronunciation dictionary and phone list automatically a system was developed.

On this work as stated a database comprised of 3125 utterances for training and 349 sentences for testing were used. Those data are pre-processed in line with the requirements of the of CMU Sphinx toolkit.

Furthermore in this work an attempted has made to build Spontaneous Speech recognizer. In order to support the Acoustic Models, a tri-gram Language Model was also constructed. In addition, pronunciation dictionary is prepared and used as an input. In addition to the CI model and CD based speech recognizer was developed. Accordingly performance tests was also conducted using the test data.

The best result was 36.83% of word accuracy and 13.76% sentence level accuracy, even though the recognizer was developed under limited resources and insufficient size of data the result obtained is a promising result, adding data size or by using the same data and modifying parameters concerning with language model and dis-fluencies, still it is possible to improve this accuracy.

Lot of tasks have been performed from data collection to data set preparation and finally a promising result was achieved. During these stages data preparation stage was the most challenging task, especially identifying sentence boundary during segmentation and transcribing the exact word as was spoken by the speaker. Because spontaneous speech is faster and more casual, many phonemes are deleted or severely shortened handling such problems were also anther challenging task.

The size of data used in this work is very low compared with databases used for other languages such as large-scale spontaneous speech database “Corpus of Spontaneous Japanese (CSJ)”. On this work training data of 510 hours long and 6.84 M words for language modeling, best Word Error Rate of 25.3% obtained [35]. In addition to data size and domain, the effect of language model on the speech recognizer can be observed from the works in [35]. So preparing large data for language modeling has avital role on increasing the performance.

The variability of spontaneous speech was also another factor. Unlike read speech spontaneous speech have different way of speaking depending on social attributes such as age, gender, education, profession, and different way sentence usages and also the speaks loudness (high tone speech) and lowness (low tone). Moreover, one speaker could speak in quite different speaking styles according to social and personal conditions in this work the data used were noise data with different speaking style. This variability requires analyses based on a large data size.

6.2 Recommendations

From the experimental results and from the point view of this study, an attempt was made to focus on investigating the possibility of developing, large vocabulary spontaneous speech recognition for Tigrigna. Depending on finings and challenges facing during the development followings are the possibilities that should have to be done for the future in order to boost the performance of the recognizer.

As stated in section 1.7.2 there is no spontaneous speech corpus prepared previously, therefore in this work an attempt has made on developing Spontaneous preparing the required data set. But it does not mean that the data set is sufficient therefore in the future researchers whom work on this area have to increasing the size of the data set, doing this helps on improving the recognizer accuracy and even to ensure the applicability of ASR. Since it is difficult to use ASR with low accuracy in a real world.

In this study the experiment was conducted with using tri-gram language model with Laplace smoothing techniques in order to remove zero probabilities and only the one with less Perplexity was selected. For future the language model has to be increased in size as well as the system has to be experimented with different smoothing techniques and increasing tri-gram to other N-grams will improve the recognizer accuracy.

One of the difficult in spontaneous speech is non-speech events (dis-fluencies), therefore handling none speech events has a big role in recognizer’s performance. In this work an attempt has been made on handling these issues, but still it is possible to handle them further by applying different techniques. One of the task one can do in order to handle these non-speech events effect is

including or removing from acoustic, lexical and language model, by studying their nature in detail and separately.

In this research a recognized (transcribed) pronunciation dictionary was used, but there are words with the same meaning but uttered differently by different speakers (such as ክልተ (two) and ኽልተ (two)), dialect and accent variations were not considered in this work. Such variations decrease the performance of the recognizer. Therefore it is better to use alternative pronunciation dictionary in order to improve recognizer performance.

As indicated on this work database used was not restricted in any domain and the data was collected from different sources whereas [10] develops dictation system for Amharic language and founds good result so it is valuable to develop domain dependent system for Tigrigna.

6.3 Contribution of the work

The main contributions of this thesis work are summarized as follows:

- spontaneous speech database was prepared that is useful for researchers who are interested on doing research on spontaneous speech recognition
- The spontaneous audio database was manually transcribed.
- A system was developed using java that helps researchers on generating pronunciation dictionary, phone list and other related tasks.
- A spontaneous speech recognizer was also developed that helps as input for next researchers.
- A trigram language model was also developed for 30931 sentences
- In this work the context dependent and independent recognizer have been developed.

References

[1] a. S. F. B.H. Juang, Automatic Recognition and Understanding of Spoken Language–A First Step Toward Natural Human Machine Communication, IEEE, 2000.

[2] M. Zhang, "Overview of speech recognition and related machine learning techniques," 2004.

[3] V. Cole R. Ward W. Zue, "Speech Recognition. Survey of the State of the Art n Human Language Technology," 1997, 1996.

[4] L. R. R. B.H. Juang, "Automatic Speech Recognition – A Brief History of the Technology," Gorgia,Atlanta, 2004.

[5] J. B.-H. Rabiner Lawrence, Fundamentals of Speech Recognition, Prentice Hall , New Jersey, 1993.

[6] A. G. Adami, "Automatic Speech Recognition: From the Beginning to the Portuguese Language".

[7] Hafte Abera, "Hidden Markov Model Based Tigrigna Speech Recognition (HMM)," 2009.

[8] Kassahun Gelan, "Continuous speech recognition for Afaan Oromoo," Addis Ababa, 2012.

[9] Adugna Deksiso, "Spontaneous Speech Recognition for Amharic Using Hidden Markov Model (HMM)," 20015.

[10] Bantegize Alemayhu, "BRANA: application of Amharic speech recognition system for dictation in judicial domain Using Hidden Markov Model (HMM)," 20015.

[11] H. W. Holmes J., Speech synthesis and recognition, vol. 317s, second, Ed., 2001.

[12] B. H. R. L. R. Juang, Hidden Markov Models for Speech Recognition. Technometrics, vol. 33, 1991.

[13] M. M. S. H. Alula T, "LAP LAMBERT Academic Publishing (February 17, 2013)," 2013.

[14] Kinfe Tadesse, "Sub-Word Based Amharic Word Recognition: An Experiment Using Hidden Markov Model (HMM).," 2002.

[15] Solomon Teferra, "Automatic Speech Recognition for Amharic," Addis Ababa, 2001.

[16] E. G. e. a. M. S. Rita Singh Ravi Mosur Ronald Rosenfeld Yitao Sun David Huggins-Daines Arthur Chan, "The Hieroglyphs: Building".

[17] "cmusphinx.sourceforge.net," [Online]. Available: http://cmusphinx.sourceforge.net/html/compare.php. [Accessed 2016].

[18] M. DeGroot, Optimal statistical decisions, first, Ed., New York: McGraw-Hill, 1969, 1969.

[19] a. D. H. L.Xuedong, "Challenges in adopting speech recognition," *Communications of the ACM*, vol. 47, p. 69.

[20] H. P. l. p. (. a. f. s. Hermansky, "The Journal," *87*, pp. 1738-1752, 1990.

[21] M. S. Z. F. Q. a. T. H. Bo Xu Bing, "Speaker-independent Dictation of Chinese Speech with 32K Vocabulary, Speech Research Group, National Lab of Pattern Recognition Institute of Automation, Chinese Academy of Sciences, Beijing, CF".

[22] S. Berhanu, "Isolated Amharic Consonant-Vowel (CV) Syllable Recognition: An experiment using the Hidden Markov Model.," 2001.

[23] Z. Seifu, "Hidden Markov Model Based Large Vocabulary, Speaker Independent, Continuous Amharic Speech Recognition," 2003, 2003.

[24] D. J. \. J. H. Martin, Speech and language processing, First, Ed., Prentice Hall.

[25] B. R. J. H. a. R. S. Rita Singh, "Robust Group Tutorial," 2014. [Online]. Available: http://www.speech.cs.cmu.edu/sphinx/tutorial.html. [Accessed 2016].

[26] P. L. P. K. B. R. R. S. E. G. P. W. a. J. W. Willie Walker, "Sphinx-4: A Flexible Open Source Framework for Speech Recognition," Sun Microsystems, Inc. Mountain View, CA,, USA, 2004.

[27] Nazareth Amlesom Kifle, "Tigrigna".

[28] T. Daniel, Modern Tigrigna Grammer.

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

[30] R. A. Mohammad Abushariah, "Arabic Speaker-Independent Continuous Automatic Speech Recognition Based on a Phonetically Rich and Balanced Speech Corpus," *The International Arab Journal of Information Technolog,* Vols. 9, No. 1,, January 2012.

[31] "Spontaneous Speech: How People Really Talk and Why Engineers Should Care," Menlo Park, CA 94704, USA.

[32] Olsen, "Fundamentals of Linguistic Analysis".

[33] . Daben Liu, "Improvements in Spontaneous Speech Recognition," Cambridge, MA 02138.

[34] M. H. S. E. Butzberger J., "Spontaneous Speech Effects in Large Vocabulary Speech Recognition Applications:," SRI International Speech Research and Technology Program, Menlo Park,, CA 94025 .

[35] S. Furui, "recent progress in corpus-based spontaneous speech recognition," IEICE Trans. Inf. & Syst., 2005.

Appendices

7. Appendix A Tigrigna Phonemes and Their Representation

Tigrigna font(Consonants)	Representation in this paper
ሀ	h
ለ	l
ሐ	ḥh
መ	m
ረ	r
ሰ	s
ሸ	sh
ቀ	q
ቐ	qq
በ	b
ተ	t
ቸ	ch
ነ	n
ኘ	gn
ከ	k
ኸ	kk
ወ	w
ዐ	ah
ዘ	z
ዠ	zz
የ	y

ድ	d
ጅ	j
ግ	g
ጥ	x
ጭ	c
ፅ	ts
ፍ	f
ፕ	p
ቭ	v
ኵ	kwie
ኹ	khwie
ቊ	kwie
ቈ	qhwie
ጕ	gwie

List of phones

+FP+	k
+BR+	khw
+LP+	kk
+HES+	kw
+THC+	
+OTH+	l
+ THC +	m
+NOISE+	n
SIL	o
a	p
ah	pp
ai	q
b	qq
	qw

c ch d e f g gn gw h hh i ie j	r s sh t ts u v w x y z zz

8. Appendix B: Sample Pronunciation dictionary (Phone-based list of dictionary taken form main dictionary)

ሀ	h e
ሀመይ	h e m e y
ሀም	h e m
ሀምልገለፁለይ	h e m l g e l e ts u l e y
ሀምቲ	h e m t i
ሀምኡ	h e m u
ሀምዚ	h e m z i
ሀርዝ	h e r z
ሀቴም	h e t ie m
ሀዚ	h e z i
ሀጋጥም	h e g a x m
ሂስ	h i s
ሂቡ	h i b u
ሂቡኒ	h i b u n i
ሂብና	h i b n a
ሂቦም	h i b o m
ሂቦምና	h i b o m n a
ሂቦምኺ	h i b o m kk i
ሂንት	h i n t
ሂወተይ	h i w e t e y
ሂወት	h i w e t
ሂወትም	h i w e t m
ሂወትና	h i w e t n a

ሂወትኪ	h i w e t k i
ሂወቶም	h i w e t o m
ሂድን	h i d n
ሂድንጅሪ	h i d n j r i

9. Appendix C: Configuration script for sphinx trainer (Take from Sphinxtrain.config)

```
# Configuration script for sphinx trainer              -*-mode:Perl-*-
$CFG_VERBOSE = 1;                    # Determines how much goes to the screen.
# These are filled in at configuration time
$CFG_DB_NAME = "TigSpeech";
# Experiment name, will be used to name model files and log files
$CFG_EXPTNAME = "$CFG_DB_NAME";
# Directory containing SphinxTrain binaries
$CFG_BASE_DIR = "/home/user/Desktop/TigSpeech";
$CFG_SPHINXTRAIN_DIR = "/usr/local/lib/sphinxtrain";
$CFG_BIN_DIR = "/usr/local/libexec/sphinxtrain";
$CFG_SCRIPT_DIR = "/usr/local/lib/sphinxtrain/scripts";
# Audio waveform and feature file information
$CFG_WAVFILES_DIR = "$CFG_BASE_DIR/wav";
$CFG_WAVFILE_EXTENSION = 'wav';
$CFG_WAVFILE_TYPE = 'mswav'; # one of nist, mswav, raw
$CFG_FEATFILES_DIR = "$CFG_BASE_DIR/feat";
$CFG_FEATFILE_EXTENSION = 'mfc';
# Feature extraction parameters
$CFG_WAVFILE_SRATE = 16000.0;
$CFG_NUM_FILT = 25; # For wideband speech it's 25, for telephone 8khz reasonable value is 15
$CFG_LO_FILT = 130; # For telephone 8kHz speech value is 200
$CFG_HI_FILT = 6800; # For telephone 8kHz speech value is 3500
$CFG_TRANSFORM = "dct"; # Previously legacy transform is used, but dct is more accurate
$CFG_LIFTER = "22"; # Cepstrum lifter is smoothing to improve recognition
$CFG_VECTOR_LENGTH = 13; # 13 is usually enough
```

```
$CFG_MIN_ITERATIONS = 1;  # BW Iterate at least this many times
$CFG_MAX_ITERATIONS = 10; # BW Don't iterate more than this, somethings likely wrong.
# (none/max) Type of AGC to apply to input files
$CFG_AGC = 'none';
# (current/none) Type of cepstral mean subtraction/normalization
# to apply to input files
$CFG_CMN = 'current';
# (yes/no) Normalize variance of input files to 1.0
$CFG_VARNORM = 'no';
# (yes/no) Train full covariance matrices
$CFG_FULLVAR = 'no';
# (yes/no) Use diagonals only of full covariance matrices for
# Forward-Backward evaluation (recommended if CFG_FULLVAR is yes)
$CFG_DIAGFULL = 'no';
# (yes/no) Perform vocal tract length normalization in training.  This
# will result in a "normalized" model which requires VTLN to be done
# during decoding as well.
$CFG_VTLN = 'no';
# Starting warp factor for VTLN
$CFG_VTLN_START = 0.80;
# Ending warp factor for VTLN
$CFG_VTLN_END = 1.40;
# Step size of warping factors
$CFG_VTLN_STEP = 0.05;
# Directory to write queue manager logs to
$CFG_QMGR_DIR = "$CFG_BASE_DIR/qmanager";
# Directory to write training logs to
$CFG_LOG_DIR = "$CFG_BASE_DIR/logdir";
# Directory for re-estimation counts
```

```
$CFG_BWACCUM_DIR = "$CFG_BASE_DIR/bwaccumdir";
# Directory to write model parameter files to
$CFG_MODEL_DIR = "$CFG_BASE_DIR/model_parameters";
# Directory containing transcripts and control files for
# speaker-adaptive training
$CFG_LIST_DIR = "$CFG_BASE_DIR/etc";
# Decoding variables for MMIE training
$CFG_LANGUAGEWEIGHT = "11.5";
$CFG_BEAMWIDTH      = "1e-100";
$CFG_WORDBEAM       = "1e-80";
$CFG_LANGUAGEMODEL  = "$CFG_LIST_DIR/$CFG_DB_NAME.lm.DMP";
$CFG_WORDPENALTY    = "0.2";
# Lattice pruning variables
$CFG_ABEAM            = "1e-50";
$CFG_NBEAM            = "1e-10";
$CFG_PRUNED_DENLAT_DIR  = "$CFG_BASE_DIR/pruned_denlat";
# MMIE training related variables
$CFG_MMIE = "no";
$CFG_MMIE_MAX_ITERATIONS = 5;
$CFG_LATTICE_DIR = "$CFG_BASE_DIR/lattice";
$CFG_MMIE_TYPE   = "rand"; # Valid values are "rand", "best" or "ci"
$CFG_MMIE_CONSTE = "3.0";
$CFG_NUMLAT_DIR  = "$CFG_BASE_DIR/numlat";
$CFG_DENLAT_DIR  = "$CFG_BASE_DIR/denlat";
# Variables used in main training of models
$CFG_DICTIONARY     = "$CFG_LIST_DIR/$CFG_DB_NAME.dic";
$CFG_RAWPHONEFILE   = "$CFG_LIST_DIR/$CFG_DB_NAME.phone";
$CFG_FILLERDICT     = "$CFG_LIST_DIR/$CFG_DB_NAME.filler";
$CFG_LISTOFFILES    = "$CFG_LIST_DIR/${CFG_DB_NAME}_train.fileids";
```

```
$CFG_TRANSCRIPTFILE                                                    =
"$CFG_LIST_DIR/${CFG_DB_NAME}_train.transcription";
$CFG_FEATPARAMS     = "$CFG_LIST_DIR/feat.params";
# Variables used in characterizing models
$CFG_HMM_TYPE = '.cont.'; # Sphinx 4, PocketSphinx
#$CFG_HMM_TYPE  = '.semi.'; # PocketSphinx
#$CFG_HMM_TYPE  = '.ptm.'; # PocketSphinx (larger data sets)

if (($CFG_HMM_TYPE ne ".semi.")
    and ($CFG_HMM_TYPE ne ".ptm.")
    and ($CFG_HMM_TYPE ne ".cont.")) {
  die "Please choose one CFG_HMM_TYPE out of '.cont.', '.ptm.', or '.semi.', " .
    "currently $CFG_HMM_TYPE\n";
}
# This configuration is fastest and best for most acoustic models in
# PocketSphinx and Sphinx-III.  See below for Sphinx-II.
$CFG_STATESPERHMM = 5;
$CFG_SKIPSTATE = 'no';
if ($CFG_HMM_TYPE eq '.semi.') {
  $CFG_DIRLABEL = 'semi';
# Four stream features for PocketSphinx
  $CFG_FEATURE = "s2_4x";
  $CFG_NUM_STREAMS = 4;
  $CFG_INITIAL_NUM_DENSITIES = 256;
  $CFG_FINAL_NUM_DENSITIES = 256;
  die "For semi continuous models, the initial and final models have the same density"
    if ($CFG_INITIAL_NUM_DENSITIES != $CFG_FINAL_NUM_DENSITIES);
} elsif ($CFG_HMM_TYPE eq '.ptm.') {
  $CFG_DIRLABEL = 'ptm';
# Four stream features for PocketSphinx
```

```
  $CFG_FEATURE = "s2_4x";
  $CFG_NUM_STREAMS = 4;
  $CFG_INITIAL_NUM_DENSITIES = 64;
  $CFG_FINAL_NUM_DENSITIES = 64;
  die "For phonetically tied models, the initial and final models have the same density"
    if ($CFG_INITIAL_NUM_DENSITIES != $CFG_FINAL_NUM_DENSITIES);
} elsif ($CFG_HMM_TYPE eq '.cont.') {
  $CFG_DIRLABEL = 'cont';
# Single stream features - Sphinx 3
  $CFG_FEATURE = "1s_c_d_dd";
  $CFG_NUM_STREAMS = 1;
  $CFG_INITIAL_NUM_DENSITIES = 1;
  $CFG_FINAL_NUM_DENSITIES = 8;
  die "The initial has to be less than the final number of densities"
    if ($CFG_INITIAL_NUM_DENSITIES > $CFG_FINAL_NUM_DENSITIES);
}
# Number of top gaussians to score a frame. A little bit less accurate computations
# make training significantly faster. Uncomment to apply this during the training
# For good accuracy make sure you are using the same setting in decoder
# In theory this can be different for various training stages. For example 4 for
# CI stage and 16 for CD stage
# $CFG_CI_TOPN = 4;
# $CFG_CD_TOPN = 16;
# (yes/no) Train multiple-gaussian context-independent models (useful
# for alignment, use 'no' otherwise) in the models created
# specifically for forced alignment
$CFG_FALIGN_CI_MGAU = 'no';
# (yes/no) Train multiple-gaussian context-independent models (useful
# for alignment, use 'no' otherwise)
$CFG_CI_MGAU = 'no';
```

```
# (yes/no) Train context-dependent models
$CFG_CD_TRAIN = 'yes';
# Number of tied states (senones) to create in decision-tree clustering
$CFG_N_TIED_STATES = 1000;
# How many parts to run Forward-Backward estimatinon in
$CFG_NPART = 1;
# (yes/no) Train a single decision tree for all phones (actually one
# per state) (useful for grapheme-based models, use 'no' otherwise)
$CFG_CROSS_PHONE_TREES = 'no';
# Use force-aligned transcripts (if available) as input to training
$CFG_FORCEDALIGN = 'yes';
# Use a specific set of models for force alignment.  If not defined,
# context-independent models for the current experiment will be used.
$CFG_FORCE_ALIGN_MODELDIR =
"$CFG_MODEL_DIR/$CFG_EXPTNAME.falign_ci_$CFG_DIRLABEL";
# Use a specific dictionary and filler dictionary for force alignment.
# If these are not defined, a dictionary and filler dictionary will be
# created from $CFG_DICTIONARY and $CFG_FILLERDICT, with noise words
# removed from the filler dictionary and added to the dictionary (this
# is because the force alignment is not very good at inserting them)
#$CFG_FORCE_ALIGN_DICTIONARY =
"$ST::CFG_BASE_DIR/falignout$ST::CFG_EXPTNAME.falign.dict";;
#$CFG_FORCE_ALIGN_FILLERDICT =
"$ST::CFG_BASE_DIR/falignout/$ST::CFG_EXPTNAME.falign.fdict";;
# Use a particular beam width for force alignment.  The wider
# (i.e. smaller numerically) the beam, the fewer sentences will be
# rejected for bad alignment.
$CFG_FORCE_ALIGN_BEAM = 1e-60;
# Calculate an LDA/MLLT transform?
$CFG_LDA_MLLT = 'no';
```

```
# Dimensionality of LDA/MLLT output
$CFG_LDA_DIMENSION = 29;
# This is actually just a difference in log space (it doesn't make
# sense otherwise, because different feature parameters have very
# different likelihoods)
$CFG_CONVERGENCE_RATIO = 0.1;
# Queue::POSIX for multiple CPUs on a local machine
# Queue::PBS to use a PBS/TORQUE queue
$CFG_QUEUE_TYPE = "Queue";
# Name of queue to use for PBS/TORQUE
$CFG_QUEUE_NAME = "workq";
# (yes/no) Build questions for decision tree clustering automatically
$CFG_MAKE_QUESTS = "yes";
# If CFG_MAKE_QUESTS is yes, questions are written to this file.
# If CFG_MAKE_QUESTS is no, questions are read from this file.
$CFG_QUESTION_SET                                                    =
"${CFG_BASE_DIR}/model_architecture/${CFG_EXPTNAME}.tree_questions";
#$CFG_QUESTION_SET = "${CFG_BASE_DIR}/linguistic_questions";
$CFG_CP_OPERATION                                                    =
"${CFG_BASE_DIR}/model_architecture/${CFG_EXPTNAME}.cpmeanvar";
# Configuration for grapheme-to-phoneme model
$CFG_G2P_MODEL= 'no';
# Configuration script for sphinx decoder
# Variables starting with $DEC_CFG_ refer to decoder specific
# arguments, those starting with $CFG_ refer to trainer arguments,
# some of them also used by the decoder.
$DEC_CFG_VERBOSE = 1;          # Determines how much goes to the screen.
# These are filled in at configuration time
# Name of the decoding script to use (psdecode.pl or s3decode.pl, probably)
$DEC_CFG_SCRIPT = 'psdecode.pl';
```

```
$DEC_CFG_EXPTNAME = "$CFG_EXPTNAME";
$DEC_CFG_JOBNAME  = "$CFG_EXPTNAME"."_job";
# Models to use.
$DEC_CFG_MODEL_NAME                                                    =
"$CFG_EXPTNAME.cd_${CFG_DIRLABEL}_${CFG_N_TIED_STATES}";
$DEC_CFG_FEATFILES_DIR = "$CFG_BASE_DIR/feat";
$DEC_CFG_FEATFILE_EXTENSION = '.mfc';
$DEC_CFG_AGC = $CFG_AGC;
$DEC_CFG_CMN = $CFG_CMN;
$DEC_CFG_VARNORM = $CFG_VARNORM;
$DEC_CFG_QMGR_DIR = "$CFG_BASE_DIR/qmanager";
$DEC_CFG_LOG_DIR = "$CFG_BASE_DIR/logdir";
$DEC_CFG_MODEL_DIR = "$CFG_MODEL_DIR";
$DEC_CFG_DICTIONARY     = "$CFG_BASE_DIR/etc/$CFG_DB_NAME.dic";
$DEC_CFG_FILLERDICT     = "$CFG_BASE_DIR/etc/$CFG_DB_NAME.filler";
$DEC_CFG_LISTOFFILES                                                   =
"$CFG_BASE_DIR/etc/${CFG_DB_NAME}_test.fileids";
$DEC_CFG_TRANSCRIPTFILE                                                =
"$CFG_BASE_DIR/etc/${CFG_DB_NAME}_test.transcription";
$DEC_CFG_RESULT_DIR     = "$CFG_BASE_DIR/result";
$DEC_CFG_PRESULT_DIR     = "$CFG_BASE_DIR/presult";
# This variables, used by the decoder, have to be user defined, and
# may affect the decoder output
$DEC_CFG_LANGUAGEMODEL=
$CFG_BASE_DIR/etc/${CFG_DB_NAME}.lm.bin";
# Or can be JSGF or FSG too, used if uncommented
# $DEC_CFG_GRAMMAR  = "$CFG_BASE_DIR/etc/${CFG_DB_NAME}.jsgf";
# $DEC_CFG_FSG  = "$CFG_BASE_DIR/etc/${CFG_DB_NAME}.fsg";
$DEC_CFG_LANGUAGEWEIGHT = "10";
$DEC_CFG_BEAMWIDTH = "1e-80";
```

```
$DEC_CFG_WORDBEAM = "1e-40";
$DEC_CFG_ALIGN = "builtin";
$DEC_CFG_NPART = 1#  Define how many pieces to split decode in
# This variable has to be defined, otherwise utils.pl will not load.
$CFG_DONE = 1;
return 1;
```

10. Appendix D: Java and Xml code

The java source code for listening to the microphone

```
FileOutputStream fos = null;
      try {
         if (rbtLive.isSelected()) {
            fos = new FileOutputStream("live Recognized word" + ".txt");
            BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(fos));
            List<String> wordList = new ArrayList<>();
            ConfigurationManager cm = new ConfigurationManager(mainwid.class.getResource("test.xml"));
            Recognizer recognizer = (Recognizer) cm.lookup("recognizer");
            recognizer.allocate();
            // start the microphone or exit if the program if this is not possible
            Microphone microphone = (Microphone) cm.lookup("microphone");
            if (!microphone.startRecording()) {
               System.out.println("Cannot start microphone.");
               recognizer.deallocate();
               System.exit(1);
            }   // loop the recognition until the program exits.
            while (true) {
               System.out.println(".\n");
               Result result = recognizer.recognize();
               if (result != null) {
                  String resultText = result.getBestFinalResultNoFiller();
                  System.out.println(".." + resultText + '\n');
                  wordList.add(resultText);
                  bw.write(resultText);
                  bw.newLine();
                  bw.flush();
               } else {
                  System.out.println("I can't hear what you said.\n");
               }

            }
         } else {
```

```
        fos = new FileOutputStream("from file Recognized word" + ".txt");
        BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(fos));
        List<String> wordList = new ArrayList<>();
        URL audioFileURL = mainwid.class.getResource("trM006-126.wav");
        URL configURL = mainwid.class.getResource("new.xml");
        ConfigurationManager cm = new ConfigurationManager(configURL);
        Recognizer recognizer = (Recognizer) cm.lookup("recognizer");
        /* allocate the resource necessary for the recognizer */
        recognizer.allocate();
        // configure the audio input for the recognizer
        AudioFileDataSource dataSource = (AudioFileDataSource) cm.lookup("audioFileDataSource");
        System.out.println("data Source is \t" + dataSource);
        dataSource.setAudioFile(audioFileURL, null);
        System.out.println("dataSource is \t" + dataSource);
        // Loop until last utterance in the audio file has been decoded, in which case the recognizer will return null.
        Result result;
        result = recognizer.recognize();
        //String resultText = result.getBestFinalResultNoFiller();
        // System.out.println(resultText);
        while (true) {
            System.out.println(".\n");
            //Result result = recognizer.recognize();
            if (result != null) {
                String resultText = result.getBestFinalResultNoFiller();
                System.out.println(".." + resultText + '\n');
                wordList.add(resultText);
                bw.write(resultText);
                bw.newLine();
                bw.flush();
            } else {
                System.out.println("I can't hear what you said.\n");
            }

        }
    }
```

```
        // TODO add your handling code here:


    } catch (FileNotFoundException ex) {
        Logger.getLogger(mainwid.class.getName()).log(Level.SEVERE, null, ex);
    } catch (IOException ex) {
        Logger.getLogger(mainwid.class.getName()).log(Level.SEVERE, null, ex);
    } finally {
        try {
            fos.close();
        } catch (IOException ex) {
            Logger.getLogger(mainwid.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}
```

The xml configuration for listening to the microphone	The xml configuration from existing audio file
<!-- ** --> <!-- The Dictionary configuration --> <!-- ** --> <component name="dictionary" type="edu.cmu.sphinx.linguist.dictionary.FastDictionary"> <property name="dictionaryPath" value="file:////home/user/Desktop/ASSRT/Model/dict/TigSpeech.dic"/> <property name="fillerPath" value="file:///home/user/Desktop/ASSRT/Model/dict/TigSpeech.filler"/ > <property name="addSilEndingPronunciation" value="false"/> <property name="wordReplacement" value="<sil>"/> <property name="unitManager" value="unitManager"/> </component> <!-- ** -- > <!-- The Language Model configuration -->	<!-- ** --> <!-- The Dictionary configuration --> <!-- ** --> <component name="dictionary" type="edu.cmu.sphinx.linguist.dictionary.FastDictionary"> <property name="dictionaryPath" value="file:////home/user/Desktop/ASSRT/Model/dict/TigSpeech.dic"/> <property name="fillerPath" value="file:////home/user/Desktop/ASSRT/Model/dict/TigSpeech.filler"/> <property name="addSilEndingPronunciation" value="false"/> <property name="wordReplacement" value="<sil>"/> <property name="unitManager" value="unitManager"/> </component> <!-- ** --> <!-- The Language Model configuration --> <!-- ** --> <component name="trigramModel"

```
   <!--
********************************************************** --
>
   <component name="trigramModel"

type="edu.cmu.sphinx.linguist.language.ngram.SimpleNGramModel">
      <property name="location"

value="file:///home/user/Desktop/ASSRT/Model/dict/TigSpeech.lm"/>
      <property name="logMath" value="logMath"/>
      <property name="dictionary" value="dictionary"/>
      <property name="maxDepth" value="3"/>
      <property name="unigramWeight" value=".7"/>
   </component>

   <!--
********************************************************** --
>
   <!-- The acoustic model configuration                    -->
   <!--
********************************************************** --
>
   <component name="wsj"

type="edu.cmu.sphinx.linguist.acoustic.tiedstate.TiedStateAcousticMod
el">
      <property name="loader" value="wsjLoader"/>
      <property name="unitManager" value="unitManager"/>
   </component>

   <component                                          name="wsjLoader"
type="edu.cmu.sphinx.linguist.acoustic.tiedstate.Sphinx3Loader">
      <property name="logMath" value="logMath"/>
      <property name="unitManager" value="unitManager"/>
      <property                                            name="location"
value="file:///home/user/Desktop/ASSRT/Model/TigSpeech"/>
   </component>
```

```
<component                                          name="epFrontEnd"
type="edu.cmu.sphinx.frontend.FrontEnd">
      <propertylist name="pipeline">
         <item>microphone </item>
         <item>dataBlocker </item>
         <item>speechClassifier </item>
         <item>speechMarker </item>
         <item>nonSpeechDataFilter </item>
         <item>preemphasizer </item>
         <item>windower </item>
         <item>fft </item>
         <item>melFilterBank </item>
         <item>dct </item>
         <item>liveCMN </item>
```

```
type="edu.cmu.sphinx.linguist.language.ngram.SimpleNGramModel">
      <property name="location"

value="file:///home/user/Desktop/ASSRT/Model/dict/TigSpeech.lm"/>
      <property name="logMath" value="logMath"/>
      <property name="dictionary" value="dictionary"/>
      <property name="maxDepth" value="3"/>
      <property name="unigramWeight" value=".7"/>
   </component>
   <!--
********************************************************** -->
   <!-- The acoustic model configuration                    -->
   <!--
********************************************************** -->
   <component name="wsj"

type="edu.cmu.sphinx.linguist.acoustic.tiedstate.TiedStateAcousticModel"
>
      <property name="loader" value="wsjLoader"/>
      <property name="unitManager" value="unitManager"/>
   </component>

   <component                                          name="wsjLoader"
type="edu.cmu.sphinx.linguist.acoustic.tiedstate.Sphinx3Loader">
      <property name="logMath" value="logMath"/>
      <property name="unitManager" value="unitManager"/>
      <property                                            name="location"
value="file:///home/user/Desktop/ASSRT/Model/TigSpeech"/>
   </component>
   <!--
********************************************************** -->
   <!-- The unit manager configuration                      -->
   <!--
********************************************************** -->

   <component name="unitManager"
         type="edu.cmu.sphinx.linguist.acoustic.UnitManager"/>
   <!—
==================================================
   <component name="mfcFrontEnd"
type="edu.cmu.sphinx.frontend.FrontEnd">
      <propertylist name="pipeline">
         <item>audioFileDataSource </item>
         <item>preemphasizer </item>
         <item>windower </item>
         <item>fft </item>
         <item>melFilterBank </item>
         <item>dct </item>
         <item>liveCMN </item>
         <item>featureExtraction </item>
      </propertylist>
   </component>
   <component name="epFrontEnd"
type="edu.cmu.sphinx.frontend.FrontEnd">
      <propertylist name="pipeline">
         <item>audioFileDataSource </item>
         <item>dataBlocker </item>
```

```
        <item>featureExtraction </item>
    </propertylist>
</component>

<component name="microphone"
         type="edu.cmu.sphinx.frontend.util.Microphone">
    <property name="closeBetweenUtterances" value="false"/>
</component>
```

```
        <item>speechClassifier </item>
        <item>speechMarker </item>
        <item>nonSpeechDataFilter </item>
        <item>preemphasizer </item>
        <item>windower </item>
        <item>fft </item>
        <item>melFilterBank </item>
        <item>dct </item>
        <item>liveCMN </item>
        <item>featureExtraction </item>
    </propertylist>
</component>
```