ADDIS ABABA UNIVERSITY

ADDIS ABABA INSTITUTE of TECHNOLOGY

SCHOOL of ELECTRICAL and COMPUTER ENGINEERING

DNN-HMM Based Isolated-Word Tigrigna Speech Recognition System

BY

Gebremedhn Gebregergs

October 2018

ADDIS ABABA UNIVERSITY

ADDIS ABABA INSTITUTE of TECHNOLOGY

DNN-HMM Based Isolated-Word Tigrigna Speech Recognition System

A thesis submitted to the School of Electrical and Computer Engineering in partial fulfillment for the Degree of Master of Science in Computer Engineering

BY

Gebremedhn Gebregergs

Advisor

Menore Tekeba

Addis Ababa, Ethiopia

October, 2018

ADDIS ABABA UNIVERSITY

ADDIS ABABA INSTITUTE OF TECHNOLOGY

DNN-HMM Based Isolated-Word Tigrigna Speech Recognition System

BY

Gebremedhn Gebregergs

Approval by Board of Examiners

Dr. Yalemzewd Negash	_____
Dean, School of Electrical Computer Engineering	Signature
Mr. Menore Tekeba	_____
Advisor	Signature
_____	_____
External Examiner	Signature
_____	_____
Internal Examiner	Signature

Department of Electrical and Computer Engineering

DECLARATION

I, the undersigned, hereby declare that this thesis is my original work performed under the supervision of Mr. Menore Tekeba, has not been presented as a thesis for a degree program in any other university and all sources of materials used for the thesis are duly acknowledged.

Menore Tekeba _____

Advisor signature

Gebremedhn Gebregergs _____

Student signature

October, 2018

ACKNOWLEDGEMENT

First and most of all, I would like to thank the Almighty for everything. Then I would like to thank my advisor Mr. Menore Tekeba for his proper guidance, constructive suggestions and technical supports. I also want to express my respect and thank to friends for their support and motivation. Finally, my heartfelt gratitude goes to my family for their love and moral support.

ABSTRACT

Automatic Speech Recognition (ASR) is a process of converting a given speech signal into corresponding textual form. An ASR system consists of Feature Extraction (FE), Acoustic Model (AM) and Decoding modules. Using state-of-the-art methods and algorithms at each of the modules improves the overall performance of the system. The objective of this thesis is therefore to develop Tigrigna ASR system using improved algorithms in the modules, the focus being on the AM part. There are previous related works on many Ethiopian languages. But the only work on Tigrigna has been done using Gaussian Mixture Model (GMM) integrated with Hidden Markov Models (HMM) for the AM part. For this work DNN-HMM has been proposed instead of GMM-HMM to improve performance of Tigrigna ASR systems.

The acoustic model is created by training of the methods using an appropriate dataset. So the dataset which consists of speech and text data has been prepared by the researcher. Since the system is an isolated-word based, the text data consists of 163 words. The words were collected from a famous Tigrigna newspaper. The corresponding speech of the words was collected from a total of 86 different speakers. Then Mel frequency Cepstrum Coefficient (MFCC) features is used to extract features from the speech data to be used for the training of the AM. Both the methods, GMM-HMM and DNN-HMM, have been implemented on the AM to compare their performance. GMM-HMM was trained first which gives the AM model and a phoneme-to-sound alignment. Once the training of GMM-HMM is done, the training of DNN-HMM is followed. The training of DNN-HMM uses the same input data that was used for the GMM-HMM. In addition to that, DNN-HMM uses the phoneme-to-audio alignment found from the GMM-HMM to be used as a target output during the training.

After the training of the AM models has been done, the system is ready for experimentation and performance evaluation. The first experiment was during the training of the AM models, for selecting best values for the training parameters. Then the overall system experiment has been done to test for performance. Two experiments were done using two different speech datasets: clean and noisy; the purpose is to compare the

performance of AM methods on both cases. On the clean speech data a recognition accuracy of 97.90% has been achieved using DNN-HMM and 97.64 using GMM-HMM. Similarly, on the noisy speech DNN-HMM has performed 75.46% and GMM-HMM 69.40%.

Key words: *Acoustic Modelling, Isolated-Word, Deep Neural Network, Gaussian Mixture Models, Hidden Markov Models*

Table of Contents

Acronyms

ASR	Automatic Speech Recognition
HMM	Hidden Markov Model
GMM	Gaussian Mixture Model
MFCC	Mel Frequency Cepstrum Coefficient
PLP	Perceptual Linear Predictive
DNN	Deep Neural network
FE	Feature Extraction
AM	Acoustic Model
LM	Language Model
LDA	Linear Discriminate Analysis
LPCC	Linear Predictive Cepstral Coefficient
PLDA	Probabilistic Linear Discriminate Analysis
DBN	Deep Belief Network
HTK	Hidden Markov Model Toolkit
CV	Consonant Vowel
ANN	Artificial Neural Network
NLP	Natural Language Processing
IPA	International Phonetic Alphabet
FFT	Fast Fourier Transform
DFT	Discrete Fourier Transform
DCT	Discrete Cosine Transform
EM	Expectation Maximization
SGD	Stochastic Gradient Descent
RBM	Restricted Boltzmann Machine
FST	Finite State Transducers
GPU	Graphical Processing Unit
CPU	Central Processing Unit
MLLR	Maximum Likelihood Linear Regression
FMLLR	feature space MLLR
SAT	Speaker Adaptive Training
DSP	Digital Signal Processing

List of Tables

List of Figures

CHAPTER ONE

1. INTRODUCTION

1.1. Background

Speech is a natural communication tool for human beings except for some that are speech disabled. Reading and Writing are the results of formal learning most of the time. That means all people cannot read and write. But anyone, including those that cannot read and write, can speak at least one language that is his native language. This situation has made scientists and engineers think of a way to use it in the modern era.

Scientists and engineers have always been amazed if they could be able to speak to machines, like computer, using human language. Making machines communicate with humans using speech makes life easier. If machines are able to interpret what humans say, they can do what humans can do quickly and accurately. As a result of this interest the so called Automatic Speech Recognition (ASR) has come into life. It is a process of converting a speech signal to a sequence of words by a means of an algorithm implemented as a computer program [1]. Even though it has an incredible application in the life of human beings, the development of a system that could speak and understand like humans is a difficult task mainly caused by the complex nature of speech. The development of ASR systems is so complex that it requires expertise in many disciplines like linguistics and computer science. It has been a challenging area of research for many years.

The history of research in ASR goes decades of years back. At the beginning of researches in this area, a system model for speech analysis and synthesis was proposed which is the basis for the later researches on speech related methods and algorithms. Early ASR systems measured formant frequencies to recognize phonemes, which are basic units of sound, from the speech signal and compared the input signal with reference pattern which had been computed from the known speech signal beforehand. This kind of recognition method remained the state-of-the-art until the systems shifted into a new statistical method called Hidden Markov Model (HMM) in the 1980's [1][2][3].This method combined with Perceptual Linear Prediction (PLP) coefficients and Mel-Frequency Cepstral Coefficients (MFCC), which are used for extracting of information from given audio signals, become state-of-the-art method of that time[3]. These HMMs are also

integrated with a probability distribution called Gaussian Mixture Models (GMMs) to effectively model a phoneme.

There have not been any other techniques that could compete with GMM-HMM based ASR systems as they are well suited for speech recognition and have continuously been improved. The introduction of Neural Networks(NNs) to the area of ASR has started to change this history even though not successful at its beginning due to inadequate hardware and learning algorithms that were not capable of training NNs with many hidden layers and large output layer with large amount of training data[1][6]. This effort has continued and the problems of hardware and algorithms has overcome by the introduction of Deep Neural networks (DNNs) consisting of many hidden layers. The effort has come to success. DNN-HMM has overthrown GMM-HMM in many dimensions. The advantage of all these efforts is that, all algorithms and methods of ASR are commonly applicable to any language.

Even though the algorithms and techniques of ASR systems are common it is required that any language needs its own ASR system implementation. Tigrigna language needs its own ASR.

Tigrigna is a Semitic language spoken in Tigray region which is in the Northern part of Ethiopia, and Eritrea. It has a total of about 6.75 million speakers. It is the second most widely spoken Semitic language next to Amharic in Ethiopia [7]. So it is a great contribution to make its speakers use an ASR system. It has been tried to develop such a system before even though it needs improvements as algorithms and methods are improved through time.

Improved ASR system is the result of the improvement of its component parts. Any ASR system is composed of Feature Extraction (FE), Acoustic Model (AM), Language model (LM), pronunciation dictionary and Decoder components [8]. The FE module compresses the input signal by extracting relevant information only, the Acoustic model takes an acoustic input from FE and gives out the most probable acoustic unit. LM gives probability of words based on its knowledge of the structure of the language. By using information from AM, LM and the dictionary the last part of the system, the decoder, generates the most likely word for the speech signal given. All these components have their own contribution towards the performance of the overall system. Especially FE, AM

and the decoder are the most important parts of the system and are the focus of most researchers in the area. The focus of this research is on the AM part for Tigrigna language, to improve it using DNN-HMM.

1.2. Problem statement

As it has been discussed in section 1.1, it is clear from the name that ASR is language dependent. That is, for example a speech recognition system for English speakers, it is specifically designed for English language. A speech recognition system developed for Tigrigna speakers, the focus of this thesis, can understand only Tigrigna language etc.

According to the review of related literatures, many similar systems have been developed for different Ethiopian languages. From the study of the literatures, one of the previous works is done for Tigrigna is by Habte [7] which focuses on the Acoustic model part. HMM integrated with GMM was used for the acoustic model of that system. But according to the literatures DNN-HMM is state-of-the-art method for acoustic modeling. In this work both methods will be tested and if the latter one is found performing better then it will be recommended for the development of future systems.

1.3. Research Questions

For this thesis, the following research questions are found important to answer.

a. Can DNN-HMM perform better compared to GMM-HMM in modeling speech acoustics?
b. Is DNN based Tigrigna speech recognition system robust to noise compared with GMM based?

1.4. Objectives

The general objective of this thesis is to develop and implement Tigrigna ASR system by focusing on the acoustic modeling using DNN-HMM.

The specific objectives of the thesis are:

- Study different speech recognition techniques, algorithms and procedures.
- Study the nature of Tigrigna language in relation to SR
- Prepare resources, like speech corpus, to be used in the development process
- Design and develop a DNN-HMM based isolated-word Tigrigna SR system.
- Evaluate the performance of the developed system
- Indicate future improvements and research directions on the area.

1.5. Methodology

The following procedure was followed to do the thesis:

- The nature of ASR technology was studied
- Review of literatures was done on the area to understand the problem and possible solutions
- Necessary tools and software (algorithms, programming languages) were identified and chosen
- Nature of Tigrigna language was studied in relation to speech recognition
- Selected algorithms and tools are studied
- An appropriate dataset was prepared
- Experimentation environment was setup
- The system is trained and tested using the training and testing data respectively.

1.6. Contributions

Most Languages of the developed countries have speech corpuses prepared and ready for any speech related researches. Unluckily, there is no such chance for Tigrigna language so far. Therefore, in this work, the dataset is prepared from scratch. The text data has been transcribed manually as carefully as possible. The corresponding speech data is collected from Tigrigna speakers. So this data, especially the text corpus, can be used for related researches may be with little modification. In addition to this, the researcher has tried to identify state-of-the art techniques for the overall speech recognition system. This is an advantage for further works in the area.

1.7. Scope and Limitations

The aim of this thesis is to identify a better method for Tigrigna ASR system. So for this purpose, the system is decided to be an isolated word and small vocabulary system. A fully featured system consists of tens of thousands of words and is capable of recognizing a continuous speech.

There is no any commercially available Tigrigna corpus. This has made the researcher to prepare it from scratch which needs more money and time that he could not afford. As a result the system is limited to be small vocabulary and isolated word even though this works for the purpose of the research, which would rather be great if it had been for large vocabulary and continuous speech.

1.8. Thesis outline

The remainder of the document is organized as follows: the next section discusses on the general aspects of speech recognition systems to introduce to the area. Chapter three reviews different works that people have done previously. Chapter four discusses the nature of Tigrigna language in the context of speech recognition system. Speech recognition techniques and algorithms selected for this thesis after the review of literatures are discussed in chapter five. The analysis, design and implementation of the system is put in chapter six. The results of this thesis are presented and discussed in chapter seven. The last chapter concludes the work and indicates future works.

CHAPTER TWO

2. AUTOMATIC SPEECH RECOGNITION OVERVIEW

Development of successful speech recognition systems require knowledge and expertise from a wide range of disciplines such as signal processing, acoustics, pattern recognition, communication and information theory, linguistics, physiology, computer science and psychology, among others [14]. Any researcher is therefore required to have a good understanding of the fundamentals of speech recognition, even though not necessarily required to have a detailed knowledge in each area. The goal of this chapter is therefore to briefly discuss the fundamental principles of speech recognition that could be used as a starting point for any researcher that wants to extend or work on it towards achieving the ultimate objective of the area.

The ultimate goal of automatic speech recognition is to make machines recognize any kind of speech of a given language as human speakers of that language do. This situation has never happened in the history of speech recognition due to the challenging nature of the area. When any such system is developed, it is done with constraints. Thus, a speech recognition system is done by imposing one or more of the following: speaker dependence, isolated words, small vocabulary and constrained grammar. The factors that caused for the dependence on these constraints are [15]: lack of sophisticated yet tractable model of speech, inadequate use of human knowledge of acoustics, phonetics and lexical access in the recognizer, lack of consistent units of speech that are trainable and relatively insensitive to context and inability to account for between speaker differences and speaker specific characteristics.

Regardless of the constraints that could be considered during the development, there is a common architecture that any system is built based on. This architecture is composed of components which are responsible for the recognition process starting from analyzing the input sound signal to the decoding step which produces the corresponding text to the input signal. There are different approaches for each of the components of such a system. Modern and successful methods have to be used at each of the components for effective functioning and improved overall recognition performance. Another important factor that can greatly affect the performance of a system is unit of recognition such as phoneme, syllable, word, etc. So selecting an appropriate unit is another challenging task.

2.1. Classes of Speech Recognition systems

Speaker Independent- A system of this kind is able to recognize any kind of speech regardless of who the speaker is. Speaker independence has been viewed as the most difficult constraint to overcome. This is because most parametric representations of speech are highly speaker dependent, and a set of reference patterns suitable for one speaker may perform poorly for another speaker. So Because of the difficulties in this approach, most systems are speaker dependent systems [15].

Speaker dependent- A system is trained by speech data of a particular speaker until a reasonable performance of that system is obtained. These kinds of systems have some applications though they have the following problems [15]: the training session is an inconvenience to the user, a large amount of processing is required before a system can be used. Typically, the user must wait many hours after speaking the training sentences, certain applications such as telephone directory assistance or banking inquiries cannot tolerate the delay of a training session, a speaker's voice may change overtime due to stress, fatigue, sickness or variations in microphone positioning, etc. Generally speaker independent systems are more flexible but less accurate because of the difficulties than speaker dependent systems.

Isolated-word- is a mode of speaking where a speaker speaks one word then stops and then speaks another word and stops, and so on. It is simpler to implement than continuous speech.

Continuous- This system is significantly more difficult than isolated-word recognition. This is as a result of word boundaries are unclear in continuous speech. In isolated-word recognition, word boundaries are known which minimizes the confusion and increases recognition accuracy. Word boundaries are difficult to find in continuous speech, for example it is hard to differentiate between the phrases, "wreck a nice speech" and "recognize speech".

Large and Small vocabularies- Vocabulary is the set of words to be recognized by the system. A large vocabulary is a vocabulary of size 1000 words or more. Even though size of a vocabulary is not taken as the best parameter for measuring difficulty of a system, there are some issues that arise with the size the vocabulary. One is, the confusability of words increases as the size of the vocabulary increases, typically from 1000 words and

beyond [15], according to researchers reports. Small vocabulary has smaller vocabulary size which may start from tens of words. With small vocabulary each word can be modelled individually, because it is reasonable to expect sufficient training for small number of words. But when the size of the vocabulary increases, it is not feasible to do the same as before because there will not be sufficient training data and storage space for each word model. The solution is to use some kind of sub-word unit that could be used for many words in common.

2.2. Speech Recognition system Architecture

The core components of any ASR system are shown in Figure 3-1

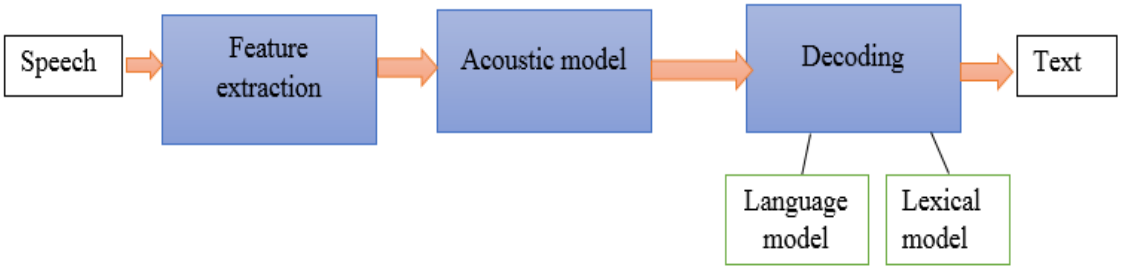


Figure 3-1 General architecture of Automatic speech recognition

Feature extraction-is the process of removing unwanted and redundant information and taking only the useful information. In another words, instead of taking the whole incoming speech signal for recognition, a compacted information that is a representative of the whole that could discriminate one signal from another is taken. If the type of information and the technique to extract it are not carefully selected the whole system may not work. That is because the classification (discrimination among different signals) is done based on the extracted feature which is believed to have the discriminating information. The features extracted are required to be invariant to changes in speaker. So to achieve these all, it needs to do some analysis. This analysis is done in two steps: temporal analysis and spectral analysis. In the first step the analysis is done on the original signal itself whereas in the second one the analysis is made on the spectral representation of the original signal. The analysis captures the dynamics of speech. The different extraction techniques used today work based on this principle.

There are different feature extraction techniques, like Linear Discriminate Analysis (LDA), Linear Predictive Cepstral Coefficients (LPCC), Perceptual Linear Predictive

Coefficients (PLP), Mel-frequency Cepstral Coefficients (MFCC), Mel scale Cepstral Analysis (MEL), Relative spectra filtering of log domain coefficients (RASTA) and First Order Derivative (DELTA) among the popular techniques. MFCC, PLP and LPC are the most widely used ones in the area of speech processing [17].

Acoustic Modeling - is used to create the relationship (alignment) between the acoustic features coming from the preceding stage and the phonemes, which it gets from the lexicon also called pronunciation dictionary, using certain methods and algorithms. In another words, the acoustic model integrates knowledge about acoustics and phonetics, takes as input the features generated from the feature extraction component, and generates an Acoustic Model (AM) score for the variable-length feature sequence. The relation (integration) is created after some training procedure, and once that relationship is created it is used as a model.

The two main issues to deal with the AM component are the variable length feature vectors and variability in the audio signals. The variability in the audio signals is caused by complicated interaction of speaker characteristics (e.g., gender, illness, or stress), speech style and rate, environment noise, side talks, channel distortion (e.g., microphone difference), dialect differences, and non-native accents. A successful speech recognition system must contend with all of these acoustic variability. This variable length feature problem is often addressed by techniques such as dynamic time warping (DTW) and HMM [21] which is capable of handling the variable nature of speech. Whenever a sound that the model knows before comes changed in some way due to the sources of variability mentioned already, the model captures this change and is able to detect it. This robustness of the model comes at a very high expense.

AM uses a computationally very high algorithms and methods. So most of the computational load and performance of the system is dependent on this part. At current time, two competing algorithms, namely Deep Neural Network (DNN) and Gaussian Mixture Model (GMM), are the commonly used methods of this part[20][21][38]. Both of these methods cannot be used together; one or the other is used at a time. Each of them are used in combination with HMM. HMM has been used most of the time in the history of ASR and is still the leading technique to handle the variable nature of speech, regardless of what is in the place of GMM and DNN which actually create the relationship

between the acoustic features and the phonemes. HMM is a kind of state machine with finite number of states, where the states represent the phonemes.

Decoding- is the last step in the process of speech recognition. The goal of all the preceding stages is realized in this step. It generates textual representation (a decoding of the acoustic input) for the corresponding speech signal. This stage needs three sources of information: the acoustic model, a pronunciation dictionary and language model (LM). LM estimates the probability of a hypothesized word sequence, or LM score, by learning the correlation between words from a (typically text) training corpora. The LM score often can be estimated more accurately if the prior knowledge about the domain or task is known. A pronunciation dictionary is a list of words with their transcription (words and the phoneme sequences they are made of.

2.3. Automatic Speech Recognition System Approaches

There are several approaches to Automatic speech recognition by machine. An overview of each approaches is given in this section. The approaches can be classified into three broad categories [1] [8] [9] [14] [19] [20]:

2.3.1. Acoustic-phonetic approach

This is on the theory of acoustic-phonetics that says there exist finite, distinctive phonetic units in spoken language and that the phonetic units are broadly characterized by a set of properties that are manifest in the speech signal or its spectrum over time. Even though the acoustic properties of phonetic units are highly variable both with speakers and with neighboring phonetic units, it is assumed that the rules governing variability are straight-forward and can readily be learned and applied in practical situations. This approach involves two steps: segmentation and labeling phase, to segment the speech signal into discrete regions where the acoustic properties of the signal are representative of one or several phonetic units and then attaching one or more phonetic labels to each segmented region according to the acoustic properties, and determining a valid word (or sequence of words) from the sequence of phonetic labels produced in the first phase.

2.3.2. Pattern recognition approach

In this approach the speech patterns are used directly without explicit feature determination and segmentation, unlike acoustic-phonetic approach. It is done in two steps: training step and comparison step. The first step enables the recognizer to know (adapt) the speech. The idea of the training step is, to have enough training data of the

intended pattern to be recognized (which can be a phoneme, word or sentence) and give this data to the training algorithm, and after adequate training, the acoustic properties of the pattern is able to be generalized. In this step the machine learns the acoustic properties which are repeatable and stable across all instances of the pattern. Then having all the possible patterns from the first step, the comparison step gets the unknown speech and compares with all the patterns and takes one pattern that best matches with the acoustic property of the speech. A speech pattern representation generally takes the form of a speech template known as template based approach or a statistical model named as stochastic approach which is equally applicable to a sound, a word, or a phrase [20].

2.3.2.1. Template based approach

In this type of approach, a reference pattern of each of the words in the dictionary is stored. Whenever an unknown speech comes, it is compared with all the reference templates and a category of a best matching template is selected. As a template of each word is constructed, it has the advantage that, errors due to segmentation or classification of smaller acoustically more variable units such as phonemes can be avoided. But it has a problem in that, every word must have its own full reference template and preparation and matching become prohibitively expensive or impractical as vocabulary size increases.

2.3.2.2. Stochastic Approach

This approach is based on the use of probabilistic models so that uncertain or incomplete information, such as confusable sounds, speaker variability, co-articulation effect and homophones words can be dealt with. This approach uses the more general modelling technique HMM, which possesses rich mathematical foundation, when it is compared with template based approach.

2.3.3. Artificial intelligence approach

This is not a new approach, its principle of operation is derived from acoustic-phonetic and pattern recognition approaches. This approach is an attempt to apply the human brain's learning procedure. That is, it mechanizes the recognition procedure according to the way a person applies its intelligence in visualizing, analyzing and finally making a decision on the measured acoustic features [14]. So in this class of recognition approach an expert system is used for segmentation and labeling so that this step can be done with methods more than that used by the acoustic-phonetic approach for its acoustic information. Specifically, this includes methods for integrating phonemic, lexical, syntactic, semantic and other related important knowledge into the expert system. Then

learning and adapting the knowledge of this expert system, and making the recognition process dynamic instead of using the static information, is the goal of this approach.

2.4. Unit of Speech Recognition

An input speech signal is not recognized as it is. Some kind of recognition unit is used so that the whole signal will be recognized as a result of recognizing such units in that signal. As has been already mentioned, HMM is an important technique of ASR systems. In this technique, it is a must that some unit of speech be used to be represented by the states of the HMM. So the question now is, what should be that unit of recognition.

Many researches and studies has been conducted on the selection of speech units and some of the identified units, each having weaknesses and strengths, are [1] [15] [19] [20]: words, phones, syllables. These are the common units used nowadays.

2.4.1. Word

It is the combination of words that phrases, sentences and other bigger units is constructed of. So technically, the recognition of speech is done by recognizing the words of that speech. This makes word as the most natural unit of speech recognition. In addition to this, word models are able to capture within word contextual effects like, the phone /t/ in ten is as expected and this phone in the word twenty may not be pronounced at all. These variations are able to be captured when using word models. So the only thing is, word models need to be adequately trained, then will give best performance. But this performance works only with small-vocabulary systems.

Using words for large vocabulary systems has many problems. Mainly in two ways, training data and storage. For each word, about 20 or more instances of that word are required for training. Then for large vocabulary system, tens of thousands of words, makes it impractical for both cases. The demand for training data gets to huge, and memory usage grows linearly with the size of data. This problem is happening due to the lack of sharing of training data among the words. Each word is for itself; that is, an instance of one word cannot be used to train for another word.

2.4.2. Phones

Phones are sub-word units, which makes possible to share training data. Training data is not a problem as is in word models by two reasons: one, the number of phones in a language is too small when compared with the number of words of that language. For

example in English there are only 50 phonemes but clearly a lot larger number of words, similarly for Tigrigna there are 29 phonemes and thousands(may be in tens of thousands) of words, etc. Two, training data can be shared among several words because it is possible that many several words contain a common phoneme(s). For example the training data for phoneme /t/ can be shared by the words ten, twenty, tea, and so on because all have that phoneme. Despite the good things, phone unit has problems.

Phone models have an inadequacy problem that they assume a phone in any context is equivalent to the same phone in any other context, which is not true in reality. A phone is affected by the phones in its immediate right and left positions, which is called co-articulation effect.

From what is discussed for word and phone models, it can be shown that they have opposite problems. Word models have the lack of generalizing, but phone models have a problem of over-generalizing.

2.4.3. Syllables

Syllables are larger units of speech than phones. They are used to avoid the co-articulatory effect that happens in phones. Even though syllables are able to protect phones at the center from the contextual effect, the beginning and end of the syllable are vulnerable to the same problem.

A more serious problem is the large number of these units. For example, there are over 20,000 syllables in English. Although this may be a small number when compared with the word models in a very large vocabulary, there are still too many parameters to reliably estimate when different units cannot share the same training data [15].

2.4.4. Triphones (Context-dependent phones)

Instead of modeling phone-in-word, these phones model phone-in-context meaning context of a phone is considered during modeling. Context in this case is the immediate left and right neighbors of a phone. A left-context dependent phone and right-context dependent phone are dependent on the left and right contexts respectively. A triphone model is a combination of both contexts, it considers the effect of both the left and right neighboring phones. Thus, if two phones have the same identity but left and right phones of that phone different, then they are considered as different triphones.

While triphone modeling is a powerful idea since it does not have the problem of co-articulation effect, it has two problems. The first problem is memory wastage. When a triphone is observed once, a model is created for it, and with a large number of triphones, the memory used could be substantial. Another problem is, triphone considers every triphone context is unique. But actually there are many phones that have similar effect on other phones. The second problem is more serious than the memory because the first problem is a matter of storage which can be solved by simply adding extra space but the latter one is a problem of concept. A solution to this is, to find instances of similar contexts and merge them together and that would result to reduced memory usage and better-trained models [15].

CHAPTER THREE

3. LITERATURE REVIEW

In this chapter different works done on automatic speech recognition are reviewed. The aim of this chapter is to review previous related works and study ASR techniques to develop a better Tigrigna speech recognition system. Since the overall quality of a speech recognition system depends on the quality of its components, each of them have to be appropriately implemented. So in this section, technologies and algorithms for each of the components are studied to develop an improved system. The review is done by organizing in to logical sections.

3.1. Feature Extraction

[19] Presented Linear Predictive coding, Mel-frequency Cepstrum Coefficient, Relative Spectral and Probabilistic Linear Discriminant Analysis as feature extraction techniques. Each of these techniques can be used to extract features in speech recognition systems but each with advantages and disadvantages. The paper concluded that Mel-frequency Cepstrum Coefficient is better than the others due to its unique behavior that makes it suitable for speech. It is designed in a form that it simulates the human auditory system.

A related work to [19] is presented in [23] that it studied different feature extraction algorithms. The study included: Linear Predictive Coding (LPC), Mel-frequency Cepstrum Coefficient (MFCC), RASTA filtering and Probabilistic Linear Discriminate Analysis (PLDA). The paper examines the characteristics, advantages and disadvantages of each of these algorithms in the context of speech recognition. The result of the comparison revealed that MFCC is best suited to speech feature extraction for better recognition.

The most famous algorithms out of the many possible implementations of automatic speech recognition are presented in [24]. From this study it is found that the most widely used feature extraction technique is MFCC.

An application of Deep Neural Networks to the acoustic modeling of a speech recognition system is demonstrated in [16]. To do this work, MFCC was used for the extracting of the speech features. The result of using these techniques gives to a good performance of the system, according to the result reported in the paper.

3.2. Acoustic modelling

The development of isolated speech recognition system by applying DNN for the acoustic model is presented in [16]. In this work, a Deep Belief Network (DBN) pre-training is used to initialize the weights of the DNN. After the weights of the DNN are being initializing by the DBN, the network is fine-tuned using back propagation algorithm.

A view of four research groups from the University of Toronto, Microsoft Research (MSR), Google and IBM Research, on a progress of methods for the acoustic modeling part is provided in [6]. Particularly, the relative advantages of DNN and GMM to the model are discussed in this paper. Both are used to determine how well each state of each HMM fits a frame or a short window of frames of coefficients that represents the acoustic input where HMM is used to deal with the temporal variability of speech. According to this study, it is shown that using the capability of today's powerful computers, and new and efficient algorithms like Deep Belief pre-training algorithms, DNN outperforms GMM.

An automatic speech recognition for speech impaired is presented in [25], by applying both GMM and DNN for the acoustic modeling part of the system. This work uses a small database of speech records due to the nature of the problem. This is because there are only a few people to get this kind of data from. In addition to this, the speech gathered is heterogeneous as the speakers of this kind speak very different from each other. The speech recording is done on a total of 15 speakers of which 6 are females and 9 are males. A speech of 3 hours duration is recorded from each speaker. One third of the total speech database is an impaired speech. A performance comparison is done between the architectures DNN-HMM and GMM-HMM using the prepared database. From the result of the comparison it is found that DNN-HMM outperforms GMM-HMM. DNN-HMM architecture improved the recognition word error rate by 13% from that of the GMM-HMM's.

A method to verify a pronunciation made by children is presented in [26]. The speech corpus for the system is prepared carefully. Two kinds of speech corpus are prepared. The first one from normal children. Only good utterances are taken from this kind. The training and testing of the model is done with 880 and 110 students respectively. Similarly, in the second kind which is from children of disordered speech, the corpus is

collected from 41 children each pronouncing 90 isolated words. 5 and 30 children are used for training and testing the model respectively. This work has used the two modelling approaches, GMM-HMM and DNN-HMM to compare them. The result of the comparison showed that for GMM-HMM a phoneme error rate of 37% and 43% for disordered and normal children is achieved respectively. For the DNN-HMM the error is found to be 21% for normal and 36% for disordered ones. Thus it can be concluded that DNN-HMM outperformed GMM-HMM.

To conclude the section, it can be shown that DNN-HMM is having a better performance over GMM-HMM. This has motivated the researcher to take these two techniques and try them on Tigrigna language.

3.3. ASR system approaches

Different approaches to ASR system are studied in [1]. The presented approaches in this paper are: acoustic-phonetic approach, pattern recognition approach, which includes template based and statistical approach, and artificial neural network approach. The most famous of all the approaches, according to this study, is the pattern recognition approach, particularly the statistical one, and that is because of HMM is in it.

A study of modern techniques for the acoustic modeling is done in [6]. As it has been discussed in section 2.3, it has mainly focused on two competing methods, GMM and DNN, which finally reported as DNN is better. But regardless of which is used, GMM or DNN or something else, HMM is being used in all. This is to mean the approach used is a pattern recognition. Similarly, [7] [16] [26] have applied the pattern recognition approach in that they have used HMM.

3.4. Unit of recognition

As it has been discussed, selecting an appropriate unit of speech recognition is an important step to the overall performance of the system. There are different recognition units where one is appropriate for one task and the other may not be, that is each of them have their own areas of application.

Different speech recognition units are presented in [20] with the advantages and disadvantages they have. Word based, phone based (context-dependent and context-independent) and syllable based are the most commonly used units according to this

study. [1][15][19] Have discussed in this area and all of these studies including [20] made a common conclusion. All of these studies concluded that, word based is good for small vocabulary systems but inappropriate for large vocabulary due to lack of adequate training data and the demand for huge storage space, context-independent based approach minimizes the demand for training data but it has a problem called co-articulation as a result of not considering for the effect of contexts, context-dependent based also called triphone unit has the same advantage as the independent one but in addition to that it avoids the co-articulation problem by considering the effect of the neighboring phones (phones in the immediate left and right positions) but there is also a problem in this; it assumes as if every triphone is unique which is unrealistic resulting to use more storage space. Syllable based units are at the middle of word based and phone based. They are affected by the problems that comes from both of its baselines, word and phone based, co-articulation effect and larger training data problems. To conclude, triphone is appropriate for large vocabulary systems.

[22] Investigated ways of improving an ASR system, particularly in the acoustic modeling part, and it has applied triphone models for this purpose. Similarly, [27] [28] have used triphone model for their purposes and this model has contributed to the good results of these works.

3.5. Related works on Tigrigna and other Ethiopian Languages

There are ASR systems implemented for many Ethiopian languages, including Tigrigna. This section is a review of such systems to study the approaches, methods and algorithms applied by such systems.

Even though the history of researches in speech recognition goes to many years back, it has begun so late to apply it to languages in Ethiopia, like Tigrigna and Amharic.

Solomon [30] has opened the door working on Amharic ASR system, according to the researcher's knowledge. He has developed a syllable based Amharic ASR using Hidden Markov Model toolkit (HTK). He has used only 41 syllables out of the total 234. He has conducted the experiment for speaker dependent and speaker independent modes and reported that he found 87.68 and 72.75 recognition accuracies respectively. The researcher has indicated that this result is not satisfying that similar approaches for other

languages has proved to perform better. This unsatisfying result can be the result of problems during dataset preparation or improper usage of the implementation tools.

Kinfe [31] has extended the work of Solomon [30] in that he has tried to investigate the performance of three sub-word units, including what Solomon has used. These sub-word units are Phones (20 out of 39 phones), Triphones and 104 Syllables. According to Kinfe, a comparison of the different sub-word units revealed that the use of CV syllables has led to relatively poor performance. This is a match with findings of [34] that phone and triphone recognizers are found performing better than CV syllables.

Another work by Zegaye [32] is on a speaker independent continuous Amharic speech recognition system. Zegaye has applied both Phone and Triphone based models to develop the system. It is found that, Triphone based recognizer has performed better than Phone based does.

Hussein and Gambäck [33] developed a speaker independent continuous speech recognizers for Amharic based on an HMM/ANN hybrid approach. The recognizer was developed using a context dependent phone unit with the help of a speech recognition toolkit called CSLU. They used 5000 sentences taken from Solomon [30]. The system is evaluated using a total of 20 sentences read by 10 speakers, 2 sentences each. These speakers has also shown in the training. The best result obtained using test data was 74.28% word recognition accuracy. When the system is tested using data from speakers who have never participated for the training, its performance has got lowered from the previous. There was 4.28% word recognition accuracy reduction, since there was no usable speech corpus for the development of a large vocabulary speech recognizer.

Solomon *et al* [36] has prepared a speech corpus for Amharic. They have designed their corpus according to best practice guidelines established for other languages. Particularly they have made it to contain the elements found on standard corpora, like Wall Street Journal speech corpus, cited by them. They have two vocabulary sets, one with 5000(5K) words and the other with 20000(20K) words. The training corpus they prepared consists of 10850 sentences, read by a total of 80 speakers, 70 of them read 100 sentences each and 10 of them read 145 sentences each. The sentences are selected in such a way that all the consonant-vowel syllables (total of 233) of the languages are included. For the test and development sets, for the 5K and 20K vocabularies, they have selected 18 and 20

different sentences respectively for 20 speakers. This work is a pretty good move in the area because preparing a speech corpus is a challenging part to prepare it from scratch. But as it was mentioned by the researchers, most of the speakers in the corpus is from one dialect (from Amharic speakers in Addis Ababa), that should have included all the dialects.

Coming towards works done related to Tigrigna speech recognition, it can be said that it is still at the beginning stage. Good works has been done in Amharic comparatively, that could motivate researchers in the area for Tigrigna. So little has been done for Tigrigna so far.

Habte [7] has made a great move to developing speech recognition system for Tigrigna. He has developed an HMM based Tigrigna speech recognizer using HTK toolkit. All the dataset for the work is prepared from scratch by the researcher. He has used a total of 300 sentences out of which 200 were used for the training and the rest 50 for testing. As the researcher has concluded, the results found are encouraging. But it also indicates much work has to be done to take the work to the next level. Absence of adequate data was one of limitations of this work, which made the system not to perform better, as indicated by the researcher. Another possible reason for the less performance could be the result of not using state-of-the-art techniques, as the focus of the research was simply to see the possibility of developing Tigrigna speech recognizer, that might made him not to worry much on performance.

Another relevant work for Tigrigna was done in [35]. In this work, they have prepared a part-of-speech tagged collection of text data for the purpose of Natural Language Processing (NLP) researches. The corpus contains 4656 cleaned sentences created by rearranging a tagged corpus of 72,080. This is a promising work for Tigrigna NLP researches. This corpus can be upgraded to a corpus that could be used for speech recognition works, but it cannot be used directly as it was created for different purpose.

To conclude the chapter, many works are studied in this section that contributed for the proposal and completion of this thesis work. To the best of the researcher's knowledge, only one Tigrigna speech recognition system was developed, the one done by Habte [7], which has been observed to have some gaps. The gap are discussed in more detail in the problem statement section of chapter 1.

CHAPTER FOUR

4. PHONETICS OF TIGRIGNA AND ITS WRITING SYSTEM

Phonetics is a form of grammar which is concerned with the study of the sounds of human language. Phonetics deals primarily with the speech sound itself, including the way in which it is produced and transmitted. Every language has its own phonetic characteristics. So the focus of this chapter is to study the basics of phonetics of Tigrigna and the writing system it uses, as this is fundamental for phone based speech recognition system.

There are three important terms to know that helps for the understanding of phonetics: phone, phoneme and allophone. Phone is the smallest unit of human sound which is recognizable but not classified. Phoneme is the smallest unit of language which distinguishes meaning. Phonemes in one language are not necessarily phonemes in another. That is some phonemes may be unique to one language only. Allophone refers to the realization of a phoneme, in other words the possible phones of a phoneme. This means that a phoneme is an abstract unit. Tigrigna language has its own characteristics in terms of the phonemes it has and its writing system.

4.1. Tigrigna Language

Tigrinya (often written as Tigrigna; /tɪˈgriːnjə/; ትግርኛ təgrəññа) is an Afro-asiatic language of the Semitic branch. It is mainly spoken in Eritrea and northern Ethiopia in the Horn of Africa, with around 7 million total speakers. Tigrinya speakers in Ethiopia (known as Tigrayans; Tigrawot; feminine Tigrāweyti, male Tigraway, plural Tegaru) number around 4.5 million individuals, and are centered in the northern Tigray Region. The Tigrinya speakers in Eritrea (Tigrinyas) total roughly 2.5 million, and are concentrated in the southern and central areas. [12]

4.2. Phonetics of Tigrigna

Phoneticians describe and classify the sound of human language in terms of the following [10]: the way in which they are produced by the vocal apparatus; the physical properties of the sound wave emanating from the speaker; and the effect the sound wave has on the various parts of the ear. Out of these description methods the first one, which is called articulatory phonetics, is most important for this thesis and is discussed in this section.

Figure 4-1 shows the different parts (places) of a human body that are responsible for the speech production (articulatory) process. As indicated by numbers in the figure, the labels are: (1) Nasal cavity, (2) Hard palate, (3) Alveolar ridge, (4) Soft palate (Velum), (5) Tip of the tongue (Apex), (6) Dorsum, (7) Uvula, (8) Radix, (9) Pharynx, (10) glottis, (11) False vocal cords, (12) Vocal cord, (13) Larynx, (14) Esophagus, and (15) Trachea.

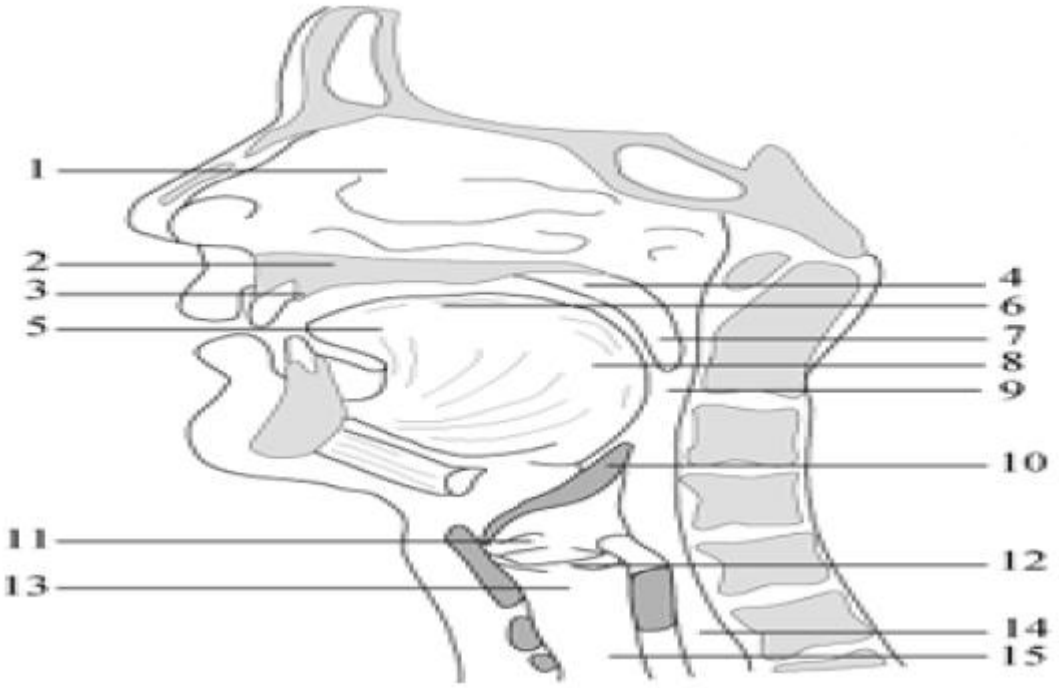


Figure 4-1 Human vocal organs adapted from [7]

In normal breathing, air is forced from the lungs, up the trachea, through the vocal cords, and out through the mouth or the nose or both. If the air is not obstructed in any way by the parts of the vocal apparatus, then no sound will be produced. Conversely, when one or more of the parts of the vocal apparatus form an obstacle in the path of the air, then a sound results [10].

Sound is produced by the rapid movement of air. Most sounds in human 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, the air passes through the larynx, commonly known as the Adam's apple or voice box. The larynx contains two small folds of muscle, the vocal folds (often referred to non-technically as the vocal cords) 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 voiceless [13]. In Tigrigna phonemes z (ዝ) and f (ፍ) are examples of voiced and unvoiced phonemes respectively. This can be proved by simply trying to pronounce them.

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 [13].

4.2.1. Consonant phonemes

There are twenty-nine consonant phonemes in Tigrigna. Among these, there is a set of ejective consonants and the usual seven-vowel system. Unlike many of the modern Ethiopian Semitic languages, Tigrinya has preserved the two pharyngeal consonants. Along with [x'], a velar or uvular ejective fricative, these phonological elements make it easy to distinguish spoken Tigrinya from closely related languages such as Amharic [12].

Another important behavior of Tigrigna phonemes is gemination. The term geminate consonant refers to, a linguistic term meaning the doubling of a consonantal sound, is meaningful in Tigrinya, i.e. it affects the meaning of words ,for example the 'd' in the word giddaf (ግደፍ) is a geminate consonant. All consonants can be geminated except the pharyngeal and laryngeals [12]. While gemination plays an important role in the morphology of the Tigrinya verb, it is normally accompanied by other marks. But there is a small number of pairs of words, which are only differentiable from each other by gemination, e.g. k'ärräbä, "he presented, he brought forth"; k'äräbä, "he came closer". The fricative sounds [x], [x^w], [x'] and [$x^{w'}$] occur as allophones that belong to the same phoneme /x/.Generally, the consonant phonemes are classified according to Manner of Voicing, places of articulation, manners of articulation [7]. In this table, for the representation of the Tigrinya sounds, a system that is common (though not universal) among linguists who work on Ethiopian Semitic languages is used, but it differs somewhat from the conventions of the International Phonetic Alphabet(IPA). So, when

the IPA symbol is different, it is indicated in square brackets. The consonant /v/ appears in parentheses because it occurs only in recent borrowings from European languages.

i. Place of Articulation

Based on the place of articulation, the category of the Tigrigna phonemes is summarized in Table 4-1 and the classes are discussed briefly as follows [13]:

Labial: Consonants whose main restriction is formed by the two lips coming together have a bilabial place of articulation.

Labiodental: consonants 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.

Alveolar: The alveolar ridge is the portion of the roof of the mouth just behind the upper teeth.

Palatal: The roof of the mouth (the palate) rises sharply from the 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 [ʔ] is made by closing the glottis (by bringing the vocal folds together).

ii. Manner of Articulation

It deals with how the restriction in airflow is made, for example whether there is a complete stoppage of air, or only a partial blockage, etc. Tigrigna consonant can be classified by the manner of articulation as Stops, Fricatives, Affricatives, Nasals, Liquids, and Semi-Vowels [7]. These classes are discussed as follows:

The Stops: A stop is a consonant in which airflow is completely blocked for a short time. This blockage is followed by an explosive sound as the air is released. The period of blockage is called the closure and the explosion is called the release [13]. There are ten stop phonemes out of which three[b(ቡ),d(ደ),g(ገ)] are voiced and the rest [p(ፕ),t(ት),k(ክ),p(ጵ), q(ቕ),ʔ (ዐ)] are unvoiced phonemes [p(ፕ),T(ት),q(ቕ)] are the glottalized counterparts of [p(ፕ),t(ት),k(ክ)].

The Fricatives: In fricatives, airflow is constricted but not cut off completely. The turbulent airflow that results from the constriction produces a characteristic 'hissing' sound [13]. Fricatives are also termed continuants. There are nine fricative phonemes out of which three[z(ዘ),ž(ዠ),'(ዐ)] are voiced and the other six [f(ፈ),S(ሸ),'p'(ጰ),š (ሠ),h(ሀ),ḥ (ሐ)] are unvoiced. In addition the velar consonants /k/ and /k'/ are pronounced differently when they appear immediately after a vowel and are not geminated. In these circumstances, /k/ is pronounced as a velar fricative. /k'/ is pronounced as a fricative, or sometimes as an affricate. This fricative or affricate is more often pronounced further back, in the uvular place of articulation, it is represented with [x'].

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

The Nasals: are made by lowering the velum and allowing air to pass into the nasal cavity [13]. There are three nasal phonemes [m(ማ), n(ነ) and ñ (ኘ)].

The Liquids: is formed by partial blockage or obstruction of air in the mouth. Liquids consist of lateral [l(ለ)]sounds and rhotic [r(ረ)]sounds. Both are voiced and dental phonemes and they occurs in all positions [7].

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 (ወ), y (የ)] in Tigrigna and they occur in all positions[7].

Approximants: In approximants, the two articulators are close together approximant but not close enough to cause turbulent airflow [13].

		Bilabial/ Labiodental	Dental	Palato-alveolar/ Palatal	Velar		Pharyngeal	Glottal
					Plain	Lab.		
Nasal		m(ማ)	n(ነ)	ñ [ɲ](ኘ)				
Stop	voiceless	p(ፐ)	t(ተ)	č [tʃ](ቸ)	k(ከ)	kw[kʷ]ኰ		’ [ʔ] (አ)
	Voiced	b(በ)	d(ደ)	ǧ [dʒ](ጀ)	g[g] (ገ)	gw[gʷ]ጐ		
	Ejective	p' [p’] (ጰ)	t'[t’] (ጠ)	č' [tʃ’](ጨ)	k'[k’] (ቀ)	kw’[kʷ’]ቈ		
Fricative	Voiceless	f(ፈ)	s(ሰ)	š [ʃ](ሸ)	(x)(ኸ)	(xw) [xʷ]ኹ	ḥ [ħ]ሐ	h(ሀ)
	voiced	(v)(ቨ)	z(ዘ)	ž [ʒ](ዠ)			‘ [ʕ] (ዐ)	
	ejective		s'[s’] (ፀ)		(x') [x’] (ቐ)	(xw')[xʷ’](ቜ)		
Approximant			l(ለ)	y[j] (የ)		w(ወ)		
Rhotic			r(ረ)					

Table 4-1: consonant phonemes of Tigrigna

4.2.2. Vowel phonemes

Like consonants, vowels can be characterized by the position of the articulators as they are made. 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. In addition Vowels differ from consonants in

that there is no noticeable obstruction in the vocal tract during their production. Air escapes in a relative unrestricted way through the mouth and/or nose [7].

The two most relevant parameters for vowels are what is called vowel height, which correlates roughly with the location of the highest part of the tongue, and the shape of the lips that is whether it is rounded or not. Vowels in which the tongue is raised toward the front are called front vowels; those in which the tongue is raised toward the back are called back vowels. Vowels in which the highest point of the tongue is comparatively high are called high vowels; vowels with mid or low values of maximum tongue height are called mid vowels or low vowels, respectively [13]. This applies to vowels of any language including Tigrigna.

In Tigrigna there are seven vowels. These are ኧ, ኡ, ኢ, ኣ, ኤ, እ and ኦ. All are voiced and oral sounds. These vowels can be found in each letters, that is, each letter in Tigrigna is not a single sound rather they are a combination of two sounds, one from vowel and one from consonant [7].

These vowels can be divided into different categories depending on how they are formulated; Front, Central or Back position of the tongue, wideness /roundness of the constriction position, and place of the tongue (high, mid or low) [11]. The vowels of Tigrigna and the class they goes into is summarized in Table 4-2.The IPA symbol is indicated in square brackets if it is different from the phoneme symbols.

Front vowels: there are two front vowels, /i/(ኢ) and /e/(ኤ), which are both unrounded. /e/ is a middle-front vowel whereas /i/ is a high front vowel. The mid-front vowel does not occur word finally but the high front vowel occurs medially as well as finally.

Central vowels: there are a total of three/ ə (እ),a (ኣ),ä (ኧ)/ which are all unrounded. Of these three /ɨ / has additional function serving as an epenthetic vowel in word-medial position to break consonant clusters that are not permitted in the language. This vowel never occurs word finally, whereas the remaining two are found both medially and finally.

Back vowels: The back vowels of Tigrigna are /u/(ኡ) and /o/(ኦ). They occur frequently in medial and final syllables. Both of these back vowels occur word medially and word finally.

	Front	Mid	Back
High	i(ኢ)	ə [ɨ] (እ)	u(ኡ)
Mid	e(ኤ)	ä [ɐ] (አ)	o(ኦ)
Low		a ኣ	

Table 4-2 vowel phonemes of Tigrigna

4.3. Writing System of Tigrigna

Tigrigna writing system is phonetic enabling anyone who wants to write Tigrigna text write that text as long as he/she can speak the language and has a good working knowledge of Ethiopic script. Unlike most languages, like English, no one needs to learn how to spell Tigrigna words, nor to see a word first written in order to know how to spell it [7].

Tigrinya is written in the Ge'ez script, originally developed for Ge'ez, also called Ethiopic. The Ge'ez script is an "abugida", each symbol representing a consonant + vowel syllable, and the symbols are organized in groups of similar symbols on the basis of both the consonant and the vowel. The writing system of Tigrigna is given in a table on Appendix A. In the table, the columns are assigned to the seven vowels of Tigrinya (and Ge'ez); they appear in the traditional order. The rows are assigned to the consonants, again in the traditional order.

For each consonant in an abugida, there is an unmarked symbol representing that consonant followed by a canonical or inherent vowel. For the Ge'ez abugida, this canonical vowel is ä, the first column in the table. However, since the pharyngeal and glottal consonants of Tigrinya (and other Ethiopian Semitic languages) cannot be followed by this vowel, the symbols in the first column in the rows for those consonants are pronounced with the vowel a, exactly as in the fourth row. These redundant symbols are falling into disuse in Tigrinya and are shown in the table colored differently (dark gray) than the others in the table. When it is necessary to represent a consonant with no following vowel, the consonant+ə form is used (the symbol in the sixth column). For example, the word ʾəntay 'what?' is written እንታይ, literally 'ə-nə-ta-yə'.

Since some of the distinctions that were apparently made in Ge'ez have been lost in Tigrinya, there are two rows of symbols each for the consonants /ħ/, /s/, and /sʾ/. In Tigray,

for /s/ and /s’/, at least one of these has fallen into disuse in Tigrinya and is now considered unnecessary. These less-used series are shown with a dark gray background in the table. The orthography does not mark gemination, so the pair of words k'ärräbä 'he approached', k'äräbä 'he was near' are both written ቀረበ. Since such minimal pairs are very rare, this presents no problem to readers of the language [12].

The entire table where these consonants and their orders are listed is known as FIDEL. Two of the Tigrigna consonants with their seven forms and orthographic symbols are given below [7]:

Orders:	1st	2nd	3rd	4th	5th	6th	7th
Transcription:	lä	lu	li	la	le	lə	lo
Orthographic symbol:	ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ
CV representation:	ልአ	ልኡ	ልኢ	ልኣ	ልኤ	ልእ	ልኦ
Transcription:	mä	mu	mi	ma	me	mə	mo
Orthographic symbol:	መ	ሙ	ሚ	ማ	ሜ	ም	ሞ
CV representation:	ምአ	ምኡ	ምኢ	ምኣ	ምኤ	ምእ	ምኦ

CHAPTER FIVE

5. SELECTED ALGORITHMS AND TOOLS

Narrowing down the domain towards the focus of the thesis, this chapter looks at the approaches, techniques, algorithms and tools used for the development of the system. Feature extraction techniques, acoustic modeling methods, decoding procedures and ASR tools are discussed in detail.

5.1. Feature extraction

As it has been said, the aim of feature extraction is to extract relevant information from an input speech signal instead of taking the signal with the unnecessary stuff. Out of the many techniques discussed in the literature review, MFCC is selected for this thesis work as it is found appropriate.

MFCC

This feature extraction approach gives a good discrimination, and the features extracted are less correlated to each other. This weak correlation between the individual features of MFCC is an advantage for the creation of statistical acoustic model.

To calculate features, acoustic observations are extracted over time frames of uniform length. Within these frames, the speech signal is assumed to be stationary. The length of these frames is typically around 25 milliseconds, for the acoustic samples in this window one multi-dimensional feature vector is calculated. The time frames are overlapping and shifted by typically 10 milliseconds. On the time window, a fast Fourier transformation (FFT) is performed, moving into the spectral domain. The working principle of this technique is based on the human auditory system.

Human ears do not perceive all frequency bands equally. This effect can be simulated with band-pass filters of non-uniform frequency band widths. Until 500 Hz, the width of the filters is 100 Hz, after that it increases logarithmically.

The filter center frequencies are defined in the so called Mel scale. The spectrum is de-correlated with a discrete cosine transformation (DCT). Of the resulting coefficients, the first coefficients carry the most significance. Therefore only the first few coefficients are selected as feature vector. The resulting features are called Mel cepstra, commonly

abbreviated as MFCC. To make it concrete, overall process of MFCC consists of frame blocking, windowing, FFT and Mel-frequency wrapping.

(i) Frame blocking - in this step the continuous speech signal is blocked into frame of N samples with adjacent frames being separated by M (M <N). The first frame consists of the first N samples. The second frame begins M samples after the first frame and overlaps it by N −M samples. Similarly, the third frame begins 2M samples after the first frame (or M samples after the second frame) and overlaps it by N − 2M samples. This process continues until all the speech is accounted for within one or more frames. Typical values for N and M are N = 256 (which is equivalent to 30 milliseconds windowing and facilitate the fast radix—2FFT and M = 100.

(ii) Windowing- the next step in the processing is to window each individual frame so as to minimize the signal discontinuities at the beginning and end of each frame. The concept here is to minimize the spectral distortion by using the window to taper the signal to zero at the beginning and end of each frame. If we define the window as w(n), 0 < n < N. Where N is the number of samples in each frame then the result of windowing is the signal:

The hamming window has the form:

$w(n) = 0.54-0.46 \cos(2\pi n/(N-1)), 0<n<N-1.$ (5.1)

(iii) Fast Fourier transform (FFT)- FFT is used for doing conversion from the spatial domain to the frequency domain. Each frame having Nm samples are converted into frequency domain. Fourier transformation is a fast algorithm to apply Discrete Fourier Transform (DFT), on the given set of Nm samples shown below:

$$\sum_{m=0}^{Nm-1} Dm.\, e^{-j2\pi km/Nm} \quad (5.2)$$

Where k=0,1,…,N_m-1.

Commonly, Dk are the combination of real and imaginary numbers thus it represents the complex numbers but, merely absolute values (frequency magnitudes) are considered to carry out further process. The obtained sequence can be interpreted as positive frequencies $0 \leq f < Fs/2$ correspond to values $0 \leq m \leq (Nm/2) - 1$, while negative frequencies $-Fs/2 < f < 0$ correspond to values $(Nm/2) + 1 \leq m \leq Nm - 1$, Fs is the sampling frequency. By calculating DFT we can obtain the magnitude spectrum.

(iv) Mel-frequency warping- is according to psychophysical studies that show human perception of the frequency contents of sound for speech signals does not follow a linear scale. The actual frequency f measured in Hz a subjective pitch is measured on a scale called the mel scale. The mel frequency scale is a linear frequency spacing below 1000 Hz and a logarithmic spacing above 1000 Hz. As a reference point the pitch of a 1 kHz tone, 40 db above the perceptual hearing threshold is defined as 1000 mels. Therefore the following approximate formula is used to compute the mels for a given frequency f in Hz.

$$\text{mel}(F) = 2595 \times \log 10(1 + f/700). \quad (5.3)$$

(v) Cepstrum- in this final step, we convert the log mel spectrum back to time. The result is called the mel frequency ceptsrum coefficients (MFFC). The cepstral representation of the speech spectrum provides a good representation of the local spectral properties of the signal for the given frame analysis. Because the mel spectrum coefficients are real number. They can be converted to the time domain using the discrete cosine transform (DCT). Therefore by denoting the result of the final step , that is mel power spectrum, coefficients as s_k , where k = 1, 2, . . . , K. Mathematical expression of MFCC, C_n, can be done as,

$$C_n = \sum_{k=1}^{K} (\log Sk) \cos[n(k - 1/2)\pi/k] \text{ , where n=1, 2, …, k.} \quad (5.4)$$

Note that we exclude the first component in c0 from DCT since it represents the mean value of the input signal which carriers little speaker specific information.

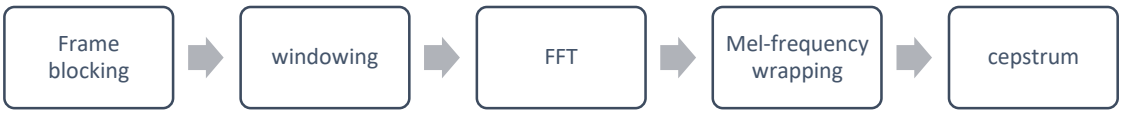


Figure 5-1 Block diagram of MFCC process

5.2. Acoustic model

Two of the competing methods in this research, GMM and DNN, integrated with HMM are used for the acoustic modeling of this system. The purpose is to implement both of them and see if the second one performs better.

5.2.1. Hidden Markov Models

Hidden Markov Model (HMM) is a well-known and widely used statistical technique of characterizing the spectral properties of frames of speech signal. The underlying assumption that made HMM applicable to speech recognition is that, the speech signal can be well characterized as a parametric random process, and that the parameters of the stochastic process can be determined in a precise and well-defined manner [14].

The idea has started with a Markov process and it later grown to HMM. The idea of HMM, HMM elements, basic problems of HMM and training ways of HMM are presented in this section.

Definition

A hidden Markov model is a collection of states connected by transitions. Each transition carries two sets of probabilities: a transition probability, which provides the probability for taking this transition, and an output probability density function (pdf), which defines the conditional probability of emitting each output symbol from a finite alphabet given that a transition is taken [15]. To summarize, HMM is a doubly embedded stochastic process with an underlying stochastic process that is not observable (it is hidden), but it can only be observed through another set of stochastic processes that produce sequence of observations [5].

Elements of HMM

An HMM is formally defined by the following elements:

a) The number of states in the model, N. Although the states of HMM are hidden as was stated, these states have some physical significance attached with them. The states are interconnected in a certain way depending on the problem they are being applied. The individual states of HMM are denoted as $S= \{S1,S2,\ldots,S_N\}$, and the state at time t as q_t.

b) Number of distinct observations per state, M. These observation symbols correspond to the physical outputs of the system being modeled. For example in a coin toss experiment, the observable outputs are sequences of heads and tails. The individual symbols are denoted as $V=\{v1,v2,\ldots, v_M\}$.

c) State transition probability distribution, $A=\{a_{ij}\}$, where

$$a_{ij}=P[q_{t+1}=S_j|q_t=S_i],\ 1<i\ ,\ j<N. \quad (5.5)$$

For all HMM types, except in the case of HMMs where any state can reach any other state in a single step and $a_{ij>0}$, $a_{ij=0}$ for one or more (i, j) pairs, where state i and state j are not directly connected.

d) The observation symbol probability distribution in state j, denoted by $B=\{b_j(k)\}$, where,
$b_j(k)=P[v_k \text{ at } t| q_t=S_j]$, $1\leq j\leq N$ and $1\leq k\leq M$. (5.6)

e) The initial state distribution, $\pi=\{\pi_i\}$, where
$\pi_{i=}P[q_{1=}Si]$, $1\leq i\leq `N$. (5.7)

Appropriate values for the parameters N, M, A, B and π must be given to generate a desired observation sequence $O=O_1O_{2\ldots}O_{T,}$ where O_t is one of the symbols from V_t and T is the total number of observations. From this one it is clear that the complete specification of the model requires the specification of N and M, specification of observation symbols, and the specification of the probability measures A, B and π. A compacted way of notation is used to have a convenient representation as: λ=(A,B, π) to give the complete specification of the model.

Basic problems of HMM

For HMMs to have life in the real world application, they have three problems that need to be solved. These problems (questions of interest) are discussed as below:

Problem a: Given a model and a sequence of observations, what is the probability that the model generated the observations? Put more clearly, given an observation sequence $O=O_1O_{2\ldots}O_T$ and the model λ=(A,B, π), how is the conditional probability P(O| λ) efficiently computed? This is called an Evaluation problem.

Problem b: given an observation sequence $O=O_1O_{2\ldots}O_T$ and the model λ, how is a corresponding optimal state sequence $Q=q1q2\ldots q_{T,}$ a state sequence that best matches the observation sequence, chosen? This problem is called a Decoding problem.

Problem c: how are the model parameters λ= (A, B, π), adjusted so that the probability of generating an observation by a given model is maximized? This is training step.

If the evaluation problem could be solved, there would be a way of scoring the match between a model and an observation sequence, which could be used for isolated-word recognition. If the decoding problem could be solved, the best matching state sequence

could be found given an observation sequence, which could be used for continuous speech recognition. Most importantly, if the learning problem could be solved, there would be a means to automatically learn the parameters given an ensemble of training data [15]. In this section, solutions to these three problems will be uncovered.

Solutions to problems of HMM

Solution to problem 1: solution to P(O| λ)

One way of solving this is enumerating every possible state sequence of length T, which is a Brute forcing way. So for an observation $O=O_1O_{2\ldots}O_T$,

Given a sample sequence $Q=q1q2\ldots q_T$

The probability of the observation sequence, O, for the state sequence, Q, is:

$$P(O|Q,\lambda) = \prod_{t=1}^{T} P(Ot|qt,\lambda) \quad (5.8)$$

then,

$$P(O|Q,\lambda)=b_{q1}(O_1).b_{q2}(O2)\ldots b_{qT}(O_T) \quad (5.9)$$

The probability of such a state sequence can be written as,

$$P(O|Q,\lambda)=\pi_{q1}a_{q1q2}a_{q2q3}\ldots a_{qT\text{-}1qT} \quad (5.10)$$

The joint probability of Q and O is then,

$$P(O|\lambda) = \sum_{all\ Q} P(O|Q,\lambda)\,P(Q|\lambda) \quad (5.11)$$

$$\sum_{q1.q2\ldots qT} \pi q1\ b1(O1)\ \ aq1q2\ bq2(O2)\ \ldots aqT-1qTbqT(OT) \quad (5.12)$$

(2T-1)Total calculations of N^T multiplications and N^T-1 additions are required to compute equation (5.8), which is computationally infeasible even for small values of N and T. For example take N=5, T=100. Therefore, there are $2*100*5^{100}{}_{=10}{}^{72}$ calculations. Therefore another more efficient procedure is used to solve the problem.

There is an efficient algorithm to solve for problem 1, called Forward-Backward algorithm. In this way, it considers a forward variable,$\alpha_t(i)$ defined by,

$$\alpha_t(i)=P(O1\ O2\ \ldots\ O_t,\ q_{t=}S_i|\lambda) \quad 5.13$$

$\alpha_t(i)$ is a probability of a partial observation sequence, $O=O_1\ O_{2\ldots}\ O_t$, until time t and state S_i at time t. $\alpha_t(i)$ can then be solved inductively as follows:

i. Initialization:

$\alpha_1(i)=\pi_i b_i(O_1),\ 1\leq i\leq N$ 5.14

This step initializes the forward probabilities as the joint probability of state S_i and initial observation O1.

ii) Induction:

$\alpha_{t+1}(j)=[\sum_{i=1}^{N}\alpha t(j)aij]bj(Ot+1), 1\leq t\leq T-1$ (5.15)

This step is the life of the forward calculation.

iii) Termination:

$P(O|\lambda)=\sum_{i=1}^{N}\alpha T(i)$ (5.16)

The above steps (i) to (iii) is called the forward and is the only one to solve the problem, it does not need the backward part which is needed during solving the third problem (training). In this algorithm the total numbers of computations are N^2T for $\alpha_t(j)$, where $1\leq t\leq T$ and $1\leq j\leq N$ unlike in the brute forcing way, which has $2TN^T$ computations.

The backward algorithm is done in a similar fashion as forward, considering a backward variable $\beta_t(i)$ defined as:

$\beta_t(i)=P(O_{t+1},O_{t+2},\ldots,O_T|q_t=S_i,\lambda)$ (5.17)

Now $\beta_t(i)$ is solved inductively the same way as was to $\alpha_t(i)$ as follows:

i) Initialization:

$\beta_T(i)=1,\ 1\leq i\leq N$ (5.18)

ii)Induction:

$\beta_t(i)=\sum_{j=1}^{N} a_{ij}bj(O_{t+1})\beta_{t+1}(j),$ (5.19)

$t=T-1,T-2,\ldots,1,1\leq i\leq N.$

Similarly as in the forward algorithm, the total numbers of computations in this are N^2T for $\beta_t(j)$, where $1\leq t\leq T$ and $1\leq j\leq N$.

Solution to problem 2

A procedure called Viterbi algorithm is used to solve problem 2.This algorithm solves the problem efficiently in that it finds a single best sequence for the observed sequence.

Viterbi Algorithm

To find the single best state sequence Q={q1q2….q_T} for a given observation sequence O={O_1 $O_{2...}$ O_T}, a quantity $\delta_t(i)$ is defined as:

$$\delta_t(i)= \max_{q1,q2,\dots,qt-1} P[q1\ q2\ \dots\ qt = i, O1O2\dots Ot|\lambda] \quad (5.20)$$

That is $\delta_t(i)$ is the best score along a single, at time t, which accounts for the first t observations and ends in state S_j. The Viterbi algorithm is stated as follows:

i) Initialization:

$$\delta_1(i)=\pi_i b_i(O_1),\ 1\le i\le N \quad (5.21)$$

$$\psi_1(i)=0 \quad (5.22)$$

ii) Recursion

$$\delta_t(j)=\max_{1\le i\le N}[\delta t-1(j)aij]bj(Ot),\ 2\le t\le T \quad (5.23)$$

$1\le i\le N$

$$\psi(i)=\operatorname{argmax}_{1\le i\le N}[\delta t-1(i)aij],\ 2\le t\le T \quad (5.24)$$

$1\le i\le N$

iii) Termination:

$$P^*=\max_{1\le i\le N}[\delta T(i)] \quad (5.25)$$

$$q^*=\max_{1\le i\le N}[\delta T(i)] \quad (5.26)$$

iv) State sequence backtracking:

$$q_t^*=\psi_{t+1}(q^*+1),t=T-1,T-2,\dots,1 \quad (5.27)$$

Solution to problem 3

This step is the most difficult one of HMM problems. There is no known way to analytically solve for the model which maximizes the probability of the observation sequence. But the model $\lambda = (A,B, \pi)$ can be chosen such that $P(O|\lambda)$ is locally maximized using an iterative such as Baum-Welch method or an equivalent method called Expectation maximization(EM), or using gradient techniques. For now Baum-Welch method is discussed as follows:

To describe the procedure for re-estimation of the HMM parameters, $\xi_t(i,j)$, probability of being in state S_i at time t and state S_j at time t+1, given the model and observation sequence, that is:

$$\xi_t(i,j)=P(q_t=S_i, q_{t+1}=S_j|O,\lambda) \quad (5.28)$$

Figure 5-2 shows the sequence of events happened leading to the conditions required by (5.24)

From the definition of forward and backward algorithms, $\xi_t(i,j)$ can be written as

$$\xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij} b_j(O_{t+1})\, \beta_{t+1}(j)}{P(O|\lambda)}$$

$$= \frac{\alpha_t(i)\, a_{ij} b_j(O_{t+1})\, \beta_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_t(i)\, a_{ij} b_j(O_{t+1})\, \beta_{t+1}(j)} \quad (5.29)$$

Let $\gamma_t(i)$ be defined as the probability of being in state S_i at time t, given the observation sequence and the model $\gamma_t(i)$ and $\xi_t(i,j)$ can be related by summing over j as:

$$\gamma t(i) = \sum_{j=1}^{N} \xi t\,(i,j) \quad (5.30)$$

Making a sum of the two over time index t,

$\sum_{t=1}^{T-1} \gamma t(i)$ =expected number of transitions from state S_i (5.31)

$\sum_{t=1}^{T-1} \xi t\,(i,j)$ =expected number of transitions from S_i to S_j (5.32)

Now based on the above equations, re-estimation formulas for the model parameters are:

$$\pi_i=\gamma_i(1), \quad \bar{a}ij = \frac{\sum_{t=1}^{T-1} \xi t\,(i,j)}{\sum_{t=1}^{T-1} \xi t\,(i,j)} \quad , \bar{b}ij = \frac{\sum_{Ot=vk}^{T} \gamma t\,(j)}{\sum_{t=1}^{T} \gamma t\,(j)}$$

After the model parameters are estimated iteratively this way and have optimal value, then the model is able to correctly identify new incoming sequences (instances of those sequences used during the training).

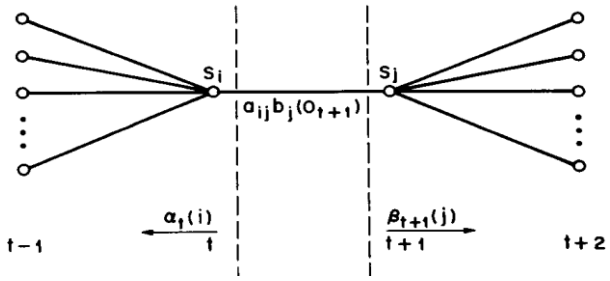


Figure 5-2 Operations for computing $\xi_t(i,j)$

Speech recognition using HMMs

Application of HMM methods usually involves the following steps [37]:

1. Define a set of L sound classes for modeling, such as phonemes or words; call the sound classes V= {v1, v2,…,v_M).

2. For each class, collect a sizable set (the training set) of labeled utterances that are known to be in the class.

3. Based on each training set, solve the estimation problem to obtain a "best" model λ_i for each class v_i (i = 1, 2, . . ., M).

4. During recognition, evaluate $P(O|\lambda_i)$(i=1,2,…,M) for the unknown utterance O and identify the speech that produced O as class v_j if $P(O|\lambda_j)$= max $P(O|\lambda_i)$, 1<i<M.

HMM implementation issues: topology

Pronunciation of a phoneme is assumed to have three phases: the first phase is when entering from the left immediate neighbor to the phoneme, called entry phase, second phase is the phone itself and third phase exiting from the phone to the right side immediate neighbor phone. This indicates that at least three states are used to represent each phone, called Triphone modelling. This behavior is shown in Figure 5-3.

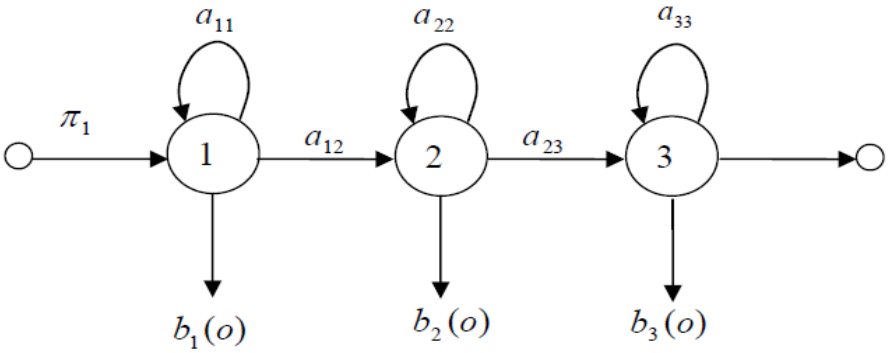


Figure 5-3 HMM topology for speech recognition

HMM implementation issue: state clustering technique

An adequate training data is not enough for one to create an accurate HMM model. Accuracy of the model can mean using more parameters per model or increasing the number of sub word models. One way to increase the accuracy without increasing the complexity of the model is to build models that share parameters. This sharing is also called as tying. Parameter tying can be done in different levels in Gaussian mixture based ASR systems. Means, co-variances, mixture components, states or transition matrices can be tied. The tying of these states is done in a certain technique.

There are two clustering ways to make the tying of the states: data driven clustering and decision tree clustering. A cluster, which is created through training, determines the states that should be tied together. In data driven clustering, the most similar states are tied. Similarity measure is the distance between these states. In Decision tree clustering (DTC), optimum decision tree is found and the states remaining in the same leaf of this tree are tied. From these clustering techniques, the latter is used for this thesis. Figure 5-4 shows way of clustering using a phoneme *aw* in different instances.

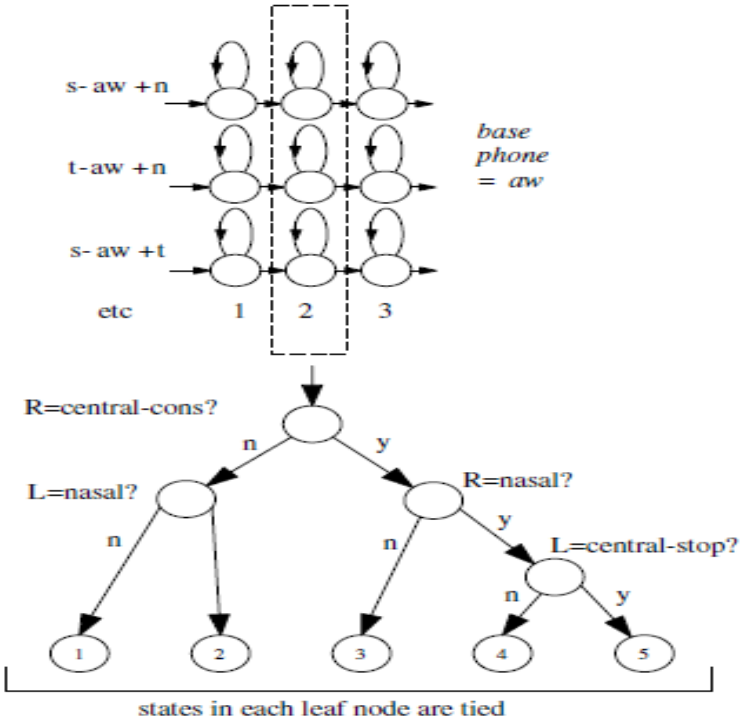


Figure 5-4 Decision tree clustering

One major property of decision trees is the capability to generalize. They are able to produce models that are not seen during acoustic training. Decision trees constructed are stored to use in the future.

Decision trees are constructed by asking a yes or no question at each of the branching nodes. Initially, all states belonging to models to be clustered are placed in the root node of the tree. As the questions are being asked, the space of states is divided into two. The number of states in a half space depends on the answer 'yes' or 'no'. Splitting the state space ends up when the optimality criterion is met. This criterion depends on two different thresholds defined by the user. The threshold can be the minimum number of states that must be in one cluster or the increase in the total likelihood of the training data on the current set of clusters. Then the states in the cluster are tied [7].

5.2.2. Gaussian Mixture Models

Gaussian distribution is one kind of probability distribution which is commonly used in many engineering and science disciplines including speech recognition. The popularity arises not only from its highly desirable computational properties, but also from its ability to approximate many naturally occurring real-world data [21].

Gaussian mixtures

A random variable x has a Gaussian distribution if it has a probability distribution function (PDF) defined as,

$$p(x) = \sqrt{\frac{r}{2\pi}}\ \exp[-\frac{r}{2}(x-\mu)^2] \tag{5.33}$$

This random variable x has mean, E(x) =μ, and variance, var(x) = $\sigma^2 = r^{-1}$.

A Gaussian-mixture random variable has a distribution of mixture of Gaussians. A scalar continuous random variable *x* has a Gaussian-mixture distribution if its PDF is specified by.

$$p(x) = \sum_{m=1}^{M} \frac{c_m}{(2\pi)^{\frac{1}{2}}\sigma_m}\ \exp\left[-\frac{1}{2}\left(\frac{x-\mu_m}{\sigma_m}\right)^2\right] \tag{5.34}$$

$$=\sum_{m=1}^{M} c_m\, N(x, \mu_m; \sigma^2{}_m),\ (-\infty < x < \infty: \sigma_m > 0; c_m > 0) \tag{5.35}$$

where the positive mixture units sum to unity: $\sum_{m=1}^{M} c_m = 1$.

The most obvious property of Gaussian mixture distribution is its multimodal one, in contrast to the unimodal property of the Gaussian distribution where *M* = 1. This makes it possible for a mixture Gaussian distribution to describe many types of physical data, including speech data, making multimodality poorly suited for a single Gaussian distribution. The multimodality in data may come from multiple underlying causes each being represented by one particular mixture component in the distribution. If such causes are identified, then the mixture distribution can be decomposed into a set of cause-dependent or context dependent component distributions.

Gaussian mixture parameter estimation

The Gaussian mixture consists of the parameters, Θ= $\{c_m, \mu_m, \Sigma_m\}$. The parameter estimation problem, also called learning, is to determine the values of these parameters from a set of data typically assumed to be drawn from the Gaussian mixture distribution. The task of parameter estimation is to learn appropriate parameters for the distribution, with the connection to the data points being represented as their membership in the individual Gaussian distributions. These parameters are estimated using the Expectation maximization algorithm (EM). The leaning through EM is done in two steps: E-step and M-step.

M-step:

$$\mathrm{Cm}^{(j+1)} = \frac{1}{N}\sum_{t=1}^{N} \mathrm{hm}^{j}(t) \quad (5.36)$$

$$\mu m^{(j+1)} = \frac{\sum_{t=1}^{N} hm^{(j)}(t)x^{(t)}}{\sum_{t=1}^{N} hm^{(j)}(t)} \quad (5.37)$$

$$\Sigma m^{(j+1)} = \frac{\sum_{t=1}^{N} hm^{(j)}(t)[x^{(t)}-\mu m^{(j)}][x^{(t)}-\mu m^{(j)}]^{T}}{\sum_{t=1}^{N} hm^{(j)}(t)} \quad (5.38)$$

The posterior probabilities (also called the membership responsibilities) are computed in the E-step:

$$hm^{(j)}(t) = \frac{Cm^{(j)}\mathcal{N}([x^{(t)};\mu m^{(j)};\Sigma m^{(j)})}{\sum_{i=1}^{n} Ci^{(j)}\mathcal{N}([x^{(t)};\mu i^{(j)};\Sigma i^{(j)})} \quad (5.39)$$

where j is an index of current estimate, x(t) is the given observation t_m mixture component and N is the number of data samples. So it means, on the basis of the current estimate for the parameters, the conditional probability for a given observation **x***(t)* being generated from mixture component t*m* is determined for each data sample point at *t* = 1, . . . , *N*. The

parameters are then updated such that the new component weights correspond to the average conditional probability and each component mean and covariance is the component specific weighted average of the mean and covariance of the entire sample set.

Gaussian Mixture Model for speech feature distribution

Gaussian mixture distribution can be used as a model for the representation of frame based speech features that is after the information about temporal order of the speech is removed. When Gaussian mixture distribution is used to represent some kind of data, it is called Gaussian Mixture Model (GMM). GMM, when used anywhere, is used for two things at the same time: modeling the data and for statistical classification. In speech recognition applications, the GMM is integrated into the doubly stochastic model of HMM to be used as an output distribution conditioned on a state [21].

5.2.3. GMM-HMM

This hybrid approach is one most common generative approach to speech recognition. GMM-HMM is a statistical model that describes two dependent random processes, an observable process which is the starting point towards identifying the spoken utterance and hidden Markov process which is not directly observable rather the path (state of HMM sequences) that might have produced the observed process is discovered using the techniques discussed earlier. A GMM-HMM is parameterized by a vector of state prior probabilities, the state transition probability matrix, and by a set of state-dependent parameters in Gaussian mixture models. As discussed earlier, when speech is modeled using GMM-HMM, each state of the model represents only part of a phone. This is the case when using context dependent modeling scheme where a single phone is represented using 3 states. But using context-dependent modeling results in the expansion of the number of states. The regularization technique discussed earlier, the state tying, handles this expansion problem. This leads to sharing of Gaussian mixture model for several states that are in the same cluster (states in the same cluster are tied).

5.2.4. Deep Neural Networks

A deep neural network (DNN) is a feed-forward, artificial neural network that has more than one layer of hidden units between its units of input and output layers [6]. A typical DNN architecture with 3 hidden layers can be shown in Figure 5-5. The hidden units use a mathematical function called activation function to process the incoming input data

from the lower layer units. Sigmoidal activation function (sometimes may be modified in some way) is used for most applications, including for this thesis. Each hidden unit, j, uses the function (5.35) to map its total input from the layer below, xj, to the scalar state, yj that it sends to the layer above.

$y_j = \text{logistic}(x_j) = \frac{1}{1+e^{-xj}}$, $x_j = bj + \sum_i yiwij$ (5.40)

A rescaled version of the sigmod function (5.35) is a hyperbolic tangent function (tanh) given as,

$$\tanh(x_j) = e^{-xj}\frac{e^{xj}-e^{-xj}}{e^{xj}+e^{-xj}} \quad (5.41)$$

where bj is the bias of unit j, i is an index over units in the layer below, and wij is a the weight on a connection to unit j from unit i in the layer below. Both of these functions have the same modeling power [21]. The output layer needs to be chosen based on the task at hand. If it is for multiclass classification, which is the case of this thesis work, output unit j converts its total input, xj , into a class probability, pj , by using the "softmax" non-linearity[6]:

$$p_j = \frac{\exp(xj)}{\sum_k \exp(xk)}, \quad (5.42)$$

where k is an index over all classes.

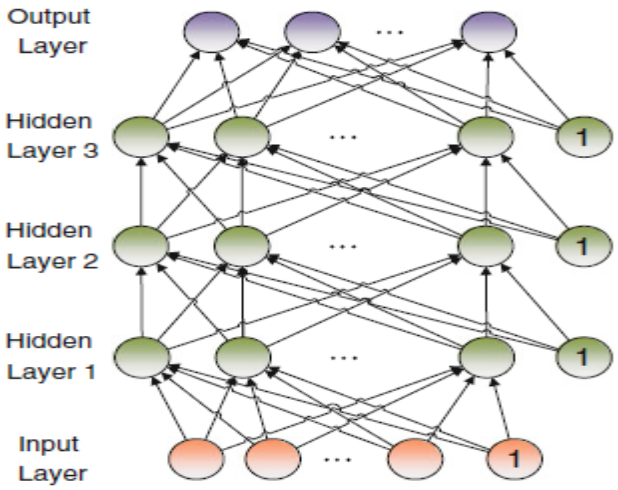


Figure 5-5: An example structure of DNN

Training DNN

DNN's are discriminatively trained by back propagating derivatives of a cost function that measures the discrepancy between the target outputs and the actual outputs produced for each training case [6]. By backpropagation it means, the errors for the hidden units of the hidden layer are determined by back-propagating the errors of the units of the output layer. When using the softmax output function, the natural cost function C is the cross-entropy between the target probabilities d and the outputs of the softmax, p:

$$C=-\sum_j dj\ log(pj), \tag{5.43}$$

For large training sets, it is typically more efficient to compute the derivatives on a small, random "mini-batch" of training cases, rather than the whole training set, before updating the weights in proportion to the gradient. This stochastic gradient descent method is further improved by using a "momentum" coefficient, $0 < \alpha < 1$, that smooths the gradient computed for mini-batch t, thereby making oscillations to be in a certain limit and speeding progress towards the goal[6]:

$$\mathrm{wij(t)} = \partial \Delta \mathrm{wij(t-1)} - \in \frac{\partial C}{\mathrm{wij(t)}} \tag{5.44}$$

Practical issues in DNN training

Model Initialization- the learning algorithms start from an initial model. Since the DNN is a highly nonlinear model and the training criterion with regard to the model parameters is nonconvex, the initial model can greatly affect the resulting model [21].

There are many heuristic tricks in initializing the DNN model. Most of these tricks are based on two considerations. First, the weights should be initialized so that each neuron operates in the linear range of the sigmoid function at the start of the learning. If weights were all very large, many neurons would saturate (will not change any more) and the gradients would be very small .When the neurons operate in the linear range, instead, the gradients are large enough (close to the maximum value of 0.25) that learning can proceed effectively. Note that the excitation value depends on both the input values and the weights, typically the sum of each input to the neuron multiplied by the weight associated with that input. When the input features are standardized, determining the initial weights can become easier. Second, it is important to initialize the parameters randomly. This is because neurons in the DNNs are symmetric and interchangeable [21]. If all the model

parameters have identical values, all the hidden layer neurons will have the same output value and detect the same feature patterns in the lower layers. Random initialization serves the purpose of symmetry breaking.

Batch size selection- Parameter update formula of training require the calculation of the empirical gradient, estimated from a batch of training samples. Batch size is the size of the training data given to the network before updating its parameters. The choice of the batch size will affect both the convergence speed and the resulting model of the network. One option to the size of training batch is the whole training data. Using this approach has a multiple of advantages [21]: convergence property of the batch training is well-known and two, it can be easily parallelized. But batch training requires a complete pass through the entire dataset before model parameters are updated, and is thus not efficient in many large scale problems even though an inefficient parallelization can be made. Another alternative for this, is a stochastic gradient descent (SGD), technique, which is sometimes referred to as online leaning in the machine learning literature. SGD updates the model parameters based on the gradients estimated from a single training sample. SGD updates the model parameters based on the gradients estimated from a single training sample. It has advantages over the batch training, batch learning will find the minimum of whatever basin the model parameters are initially in and results in a model that is highly dependent on the initial model, SGD algorithm, however, due to the noisy gradient estimation, can jump out of the poor local optima and enter a better basin. Furthermore, SGD is often much faster than batch learning especially on large datasets. This is due to two reasons. First, typically there are many similar, sometimes redundant, samples in the large datasets. Estimating the gradient by going through the whole dataset is thus waste of computation. Second, and more importantly, in the SGD training, one can make quick updates after seeing each sample. The new gradient is estimated based on the new model instead of the old model and so can move more quickly toward finding the best model.

However, batch training beats SGD in terms of parallelization. SGD is difficult to parallelize even on the same computer. In addition, SGD cannot fully converge to the local minimum, it rather fluctuates around the minimum, where the size of the fluctuation depends on the learning rate and the amplitude of the gradient estimation variance. This fluctuation, even though it can sometimes reduce overfitting, it is not desirable in many

cases. So a compromise between batch learning and the SGD algorithm is minibatch training, which estimates the gradient based on a small batch of randomly drawn training samples. It is easy to show that the gradient estimated from the minibatch is also unbiased and the variance of the estimation is smaller than that in the SGD algorithm. Minibatch training allows us to easily parallelize within the minibatch, and thus can converge faster than SGD. Since we prefer large gradient estimation variance in the early stage of the training to quickly jump out of poor local optima and smaller variance at the later stage to settle down to the minimum, we can choose smaller minibatch size initially and large ones in the later stage. In speech recognition tasks, a use of 64–256 samples in early stages and 1,024–8,096 samples in later stages is found to learn a better model [21]. An even smaller batch size is preferred at the very initial stage when a deeper network is to be trained. The minibatch size may be automatically determined based on the gradient estimation variance. Alternatively, the batch size can be determined by searching on a small subset of samples in each epoch.

*Learning Rate and stopping criteria- o*ne of the difficulties in training a DNN is selecting an appropriate learning strategy. It has been shown in theory that when the learning rate is set to $\partial = \frac{c}{t}$, where *t* is the number of samples presented and *c* is a constant, SGD converges asymptotically [21]. In practice, however, this scheme converges very slowly since the learning rate will quickly become very small.

Note that it is the combination of the learning rate and the batch size that affects the learning behavior. As noted before, smaller batch size should be used in the first several data passes and a larger batch size should be used in the later data passes. Since the batch size is variable, a corresponding per-sample learning rate is defined as, $\partial s = \frac{\partial}{Mb}$ and then model parameter update formula will be changed accordingly. With the new update formulas, the learning strategy can be determined empirically. That is, run the training for many hours and watch for the training criterion and reduce the sizes of the training batch ,learning or both in order for the quantity ∂sMb be halved until the training criteria is improved. The learning rate is divided by two and used as the initial learning rate. A large part of the training set is then used to make the training increasing the product ∂sMb by four to eight folds. This will definitely improve the training speed with no any divergence as a result of that the parameters are already adjusted.

Overfitting- overfitting is a situation happened when the network structure and parameters are beyond from what is actually required for the problem. To reduce overfitting, large weights can be penalized in proportion to their squared magnitude, or the learning can simply be terminated at the point at which performance on a held-out validation set starts getting worse [6][21].

DNN's with many hidden layers are hard to optimize. Gradient descent from a random starting point near the origin is not the best way to find a good set of weights and unless the initial scales of the weights are carefully chosen [21], the backpropagated gradients will have very different magnitudes in different layers. In addition to the optimization issues, DNNs may generalize poorly to held-out test data. DNNs with many hidden layers and many units per layer are very flexible models with a very large number of parameters. This makes them capable of modeling very complex and highly non-linear relationships between inputs and outputs. This ability is important for high-quality acoustic modeling, but it also allows them to model spurious regularities that are an accidental property of the particular examples in the training set, which can lead to a danger of overfitting. Weight penalties or early-stopping can reduce the overfitting but only by removing much of the modeling power. Very large training sets can reduce overfitting whilst preserving modeling power, but only by making training very computationally expensive. What is needed is a better method of using the information in the training set to build multiple layers of non-linear feature detectors [6]. This leads to the use of new approaches for learning deep networks.

Generative pre-training

As stated before, to solve the problems of random initialization, it is better to initialize the weights from another pre-trained network. The idea is, instead of designing feature detectors to be good for discriminating between classes, designing them to be good at modeling the structure in the input data results is more useful. The purpose is to learn one layer of feature detectors at a time with the states of the feature detectors in one layer acting as the data for training the next layer. After this generative "pre-training", the multiple layers of feature detectors can be used as a much better starting point for a discriminative "fine-tuning" phase during which backpropagation through the DNN slightly adjusts the weights found in pre-training [6][16][21]. Some of the high-level features created by the generative pre-training will be of little use for discrimination, but

others will be far more useful than the raw inputs. The generative pre-training finds a region of the weight-space that allows the discriminative fine-tuning to make rapid progress, and it also significantly reduces overfitting [6]. Deep belief network is a generative model which can be used for the pre-training step. DBN is created by a cascade of another network called Restricted Boltzmann Machine (RBM).

Restricted Boltzmann Machines- the restricted Boltzmann machine (RBM) is a stochastic generative neural network. As its name indicates, it is a variant of the Boltzmann machine. It is essentially an undirected graphical model constructed of a layer of stochastic visible neurons and a layer of stochastic hidden neurons. The visible and hidden neurons form a bipartite graph with no visible–visible or hidden–hidden connections. An example of RBM architecture is shown in Figure 5-6.The hidden neurons usually take binary values and follow Bernoulli distributions. The visible neurons may take binary or real values depending on the input types.

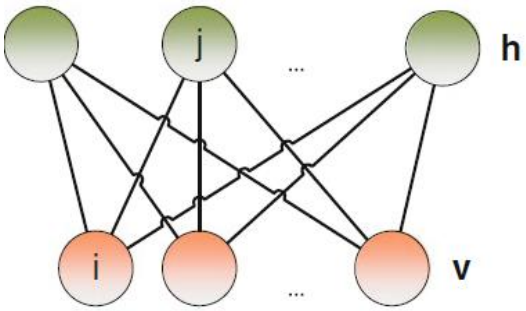


Figure 5-6 Structure of Restricted Boltzmann Machines

RBM's assign an energy value to each configuration so that dependencies between variables can be obtained with more probable configurations having a lower energy. Energy term associated with visible, *v* and hidden, *h* units is given by:

$$E(v,h)=-\sum_{i=1}^{v}\sum_{j=1}^{H} wijvihj-\sum_{i=1}^{v} bivi-\sum_{j=1}^{H} ajhj, \quad (5.45)$$

where *wij* is the weight assigned between the connections of hidden and visible units, *bi* and *aj* are their bias terms. The probability of a given configuration is given as:

$$P(v,h)=\frac{e^{-E(v,h)}}{Z}, \quad (5.46)$$

where *z* is the normalization factor, $Z=\sum_{v,h} e^{-E(v,h)}$.

As RBM is an energy-based model, training of RBM is like obtaining probability distribution. In RBM it is done by decreasing the energy of most probable regions of the state space while boosting the energy of less probable regions. It uses the following rules for updating weights and biases:

$$\Delta W^0 ij \propto \langle vihj \rangle_0 - \langle vihj \rangle_n \quad (5.47)$$

$$\Delta b^0 i \propto \langle vi \rangle 0 - \langle vi \rangle_n \quad (5.48)$$

$$\Delta b^1 i \propto \langle h1i \rangle 0 - \langle h1i \rangle_n \quad (5.49)$$

Once the RBM is trained, it comes to re-represent the data. For each data vector, **v**, a vector of expected hidden neuron activations (which equal to the probabilities) h is computed. These hidden expectations are used as training data for a new RBM. Thus each set of RBM weights can be used to extract features from the output of the previous layer. The result of the complete training of RBMs gives the initial values for all the weights of the hidden layers of a DBN, with a number of hidden layers equal to the number of RBMs. The DBN can be further fine-tuned using appropriate algorithms.

The DBN weights can be used as the initial weights in the sigmoidal DNN. This is because the conditional probability $P(\mathbf{h}|\mathbf{v})$ in the RBM has the same form as that in the DNN if the sigmoid nonlinear activation function is used.

Initializing DNN weights with generative pretraining may potentially improve the performance of the DNN on the testing set. This is due to three reasons [21]. First, the DNN is highly nonlinear and non-convex. The initialization point may greatly affect the final model especially if the batch mode training algorithm is used. Second, the generative criterion used in the pretraining stage is different from the discriminative criterion used in the backpropagation phase. Starting the backpropagation training from the generatively pretrained model thus implicitly regularizes the model. Third, since only the supervised fine-tuning phase requires labeled data, a large quantity of unlabeled data during pretraining. Experimental results have shown that generative pretraining often helps and never hurts the training of DNN [21], except that pretraining takes additional time. The generative pretraining is especially important when the training set is small.

From the preceding discussion it is shown that pretraining is a medicine for the healthy functioning of DNNs. So using pretraining is a safety during DNN training to get the

required results. In this thesis, the idea of pretraining is applied in some other way instead of implementing it directly for convenience reasons. This is explained in the implementation part.

5.2.5. Deep Neural Network-Hidden Markov Model Hybrid system

DNN-HMM architecture

The DNNs are not directly used to model speech signals since speech signals are time series signals while DNNs require fixed-size inputs. But DNNs are known for their strong classification ability. To use the strong classification ability of DNNs in speech recognition, it is necessary to find a way to handle the variable length problem in speech signals.

The nature of Hidden Markov Models (HMMs) made them suitable for handling the aforementioned situation of speech signals. Combining the capability of these two techniques (DNN -HMM) is thought to result in an improved speech recognition system, than using GMM , which is claimed to be less powered than DNN for classification, with HMM. So, testing if DNN-HMM could really improve better than GMM-HMM for Tigrigna language is the effort of this thesis. Way of integrating DNN and HMM is discussed in this section. Architecture of DNN-HMM hybrid system is shown in Figure 5-7.

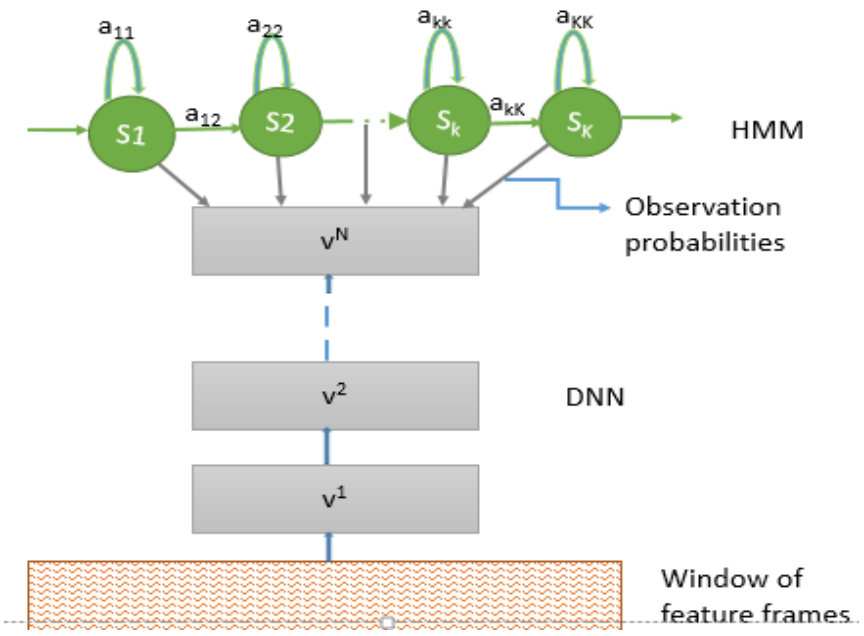


Figure 5-7 General structure of DNN-HMM system

In the DNN-HMM, a single DNN is trained to estimate the conditional state posterior probability $p(q_t = s|\mathbf{x_t})$ for all states $s \in \{s_1, s_2, \ldots, s_N\}$ unlike GMM that, a different one is used for each state. Additionally, the input to DNN is not simply a single frame of the target phone but it also includes the neighboring frames (for the neighboring phones), typically 9-13 frames of input features [21]. This is important to learn more about the target frame. The larger the frame size, the better the network output.

DNN-HMM training

DNN-HMM is trained using a supervised manner which means a target output is required for the training of the DNN. This target output is obtained from the training of GMM-HMM system which is an alignment of speech features with HMM states. In another words, DNN-HMM shares the phoneme tying structure (the decision tree cluster) and the HMM with GMM-HMM and this forces for the training of GMM-HMM model first. Clearly, the training of GMM-HMM system has to be carefully done because this affects for the performance of DNN-HMM system too.

After the DNN has been discriminatively fine-tuned, it outputs probabilities of the form p(HMMstate|AcousticInput). But to compute a Viterbi alignment or to run the forward-backward algorithm within the HMM framework we require the likelihood p(AcousticInput|HMMstate). The posterior probabilities that the DNN outputs (equation 5.37) can be converted into the scaled likelihood by dividing them by the frequencies of the HMM-states in the forced alignment that is used for fine-tuning the DNN [6]. These two methods, GMM-HMM and DNN-HMM, after properly trained, are used as the acoustic model of a speech recognition system.

5.3. Decoding

Decoding is the process to calculate which sequence of words is most likely to match to the acoustic signal represented by the feature vectors. The process of decoding is done based on three information sources: pronunciation dictionary, language model and acoustic model. Particularly, the objective of the decoder is for an observation sequence, X, to find sequence of words, $\hat{W}$.

$$\hat{W} = \arg\max_{W} P(W \mid X).$$

Using Bayes probability theorem, the above equation can be rewritten as:

$$\hat{W}=\underset{W}{\operatorname{argmax}}\ P(W|X)=\frac{P(X|W)P(W)}{P(X)} \quad (5.50)$$

$\propto P(X|W)P(W)$, since P(X) is the same for each potential sentence, because we are searching a best matching sequence of words for the same observation X.

$$\hat{W}=\underset{W}{\operatorname{argmax}}\ P(X|W)P(W)\ . \quad (5.51)$$

As it can be seen from equation (5.46), the decoding process needs an acoustic model P(X|W) and language model P(W) also called prior probability of the given word sequence.

5.4. Language model

Language model (LM) is a statistical representation of the grammar of a given languages. Formally, it is the prior probability, P(W), of the possible sequence of words for a given acoustic observation X. LMs are not always a must , for example for small vocabulary problems, whereas necessary for large vocabulary problems since the chance of occurrence of same sounding utterances increases with size of vocabulary [2], and creates a confusion for the acoustic model. LMs are used to help the acoustic model to minimize this confusion which restricts the search space by ignoring sequences that are not allowed in the language. The most widely used LMs until recently are called n-gram [2] [3] [20].

N-grams

N-grams are used to guide the search for correct word sequence by predicting the likelihood of the n^{th} word on the basis of the n−1 preceding words. The probability of occurrence of a word sequence W is calculated as:

$$P(W) = P(w1)\ P(w2|w1)\ P(w3|w1\ w2)\ldots\ldots P(w_N|w1\ w2\ \ldots\ldots w\ N\text{-}1) \quad (5.52)$$

The conditioning of the probability (word history) is limited to 2-3 as it is impractical to find occurrence of words conditioned on more than preceding 3 words [39]. So the practically used models are 2-gram also known as bi-gram and 3-gram called tri-gram. This means, in bi-gram the n^{th} word is dependent only on the preceding 2 words and in tri-gram the n^{th} word depends on the preceding 3 words. Mathematically,

Bi-gram: $P(W)= P(w1)\ P(w2|w1)\ P(w3|w1\ w2)P(w4|w2w3)\ldots P(wN|wN\text{-}2wN\text{-}1)$

Tri-gram: P(W)= P (w1) P (w2|w1) P (w3|w1 w2)P(w4|w1w2w3)…P(wN|wN-3wN-2 wN-1)

5.5. Speech recognition tools

As it has been said, the nature of speech recognition system requires developers to have expertise in many fields like linguistics, physiology, computer science, psychology, and so on. But it is obvious that, practically it is hard to have a knowledge (at least a good knowledge) of all the necessary disciplines. It would be better if researchers in this area focus only on what they want to work on, instead of wasting time on an extra tasks working on the whole system. On the other side, to demonstrate the effect of the research done, it should be tested on a complete system which enforces a researcher to implement the whole system. But thanks to engineers, many speech recognition toolkits have been developed that could handle this situation, simplifying many tasks for a researcher.

Many works have been done to develop systems, which are speech recognition toolkits, which could be used for speech recognition related researches. [45] Has identified the most widely used tools such as Hidden Markov Model toolkit (HTK), CMU sphinx, Julius and Kaldi. From the research, the performance of these tools has been tested on professional corpuses. From the reported result, Kaldi was found performing best and suitable for this purpose from the others all. The main intended use for Kaldi is acoustic modeling research. Its most competitors are found as being HTK and the RWTH ASR toolkit (RASR). The chief advantage of Kaldi compared with HTK is modern, flexible, cleanly structured code and better WFST and math support [40]. Therefore Kaldi was chosen and extensively used for the development of the system in this thesis. The next section then focusses on Kaldi and how it was used for this thesis.

5.5.1. The Kaldi speech recognition Toolkit

Kaldi is a speech recognition toolkit written in C++ and is freely available. The goal of Kaldi is to have modern and flexible code that is easy to understand, modify and extend. Kaldi is known for its specific requirements, compared with the other competitors, finite-state transducer (FST) based framework, extensive linear algebra support, and a non-restrictive license. The schematic overview of Kaldi is given in Figure 5-8. As it can be seen from the figure Basic Linear algebra Subroutines (BLAS) /Linear Algebra PACKage (LAPACK) and OpenFst are two external libraries that Kladi is highly dependent on. The rest of the library modules are grouped into two distinct halves, each depending on only

one of the external libraries. The two halves are bridged using a single module called *DecodableInterface* [40]. These two libraries together are therefore the basis to the toolkit.

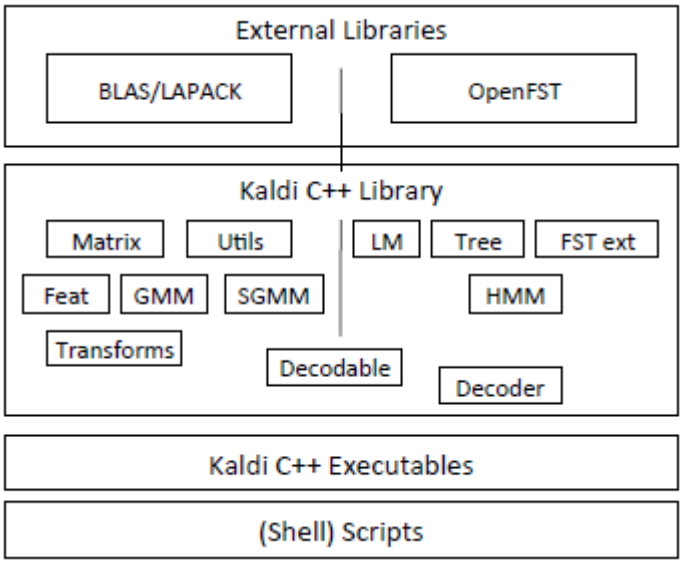


Figure 5-8: Schematic overview of Kaldi taken from [40]

The toolkit provides access to the libraries using command-line tools that are written in C++. These tools are then being used by calling them from a scripting language for building and running a speech recognizer. Kaldi has the necessary commands for each and every stage of a speech recognition system development. The discussion of the main Kaldi's commands that are used for this thesis is given in the next chapter, while this section summarizing its speech recognition components only.

Feature extraction

The feature extraction and waveform reading commands of Kaldi are implemented based on MFCC and PLP techniques. In doing this, it sets some default values while leaving available options, like minimum and maximum frequency cutoffs, that people are most likely to want to set or modify by themselves.

Acoustic Models

The aim of Kaldi for acoustic modelling is to support conventional methods and extend to new kinds of models. It supports diagonal GMM which is the conventional model and three DNN implementations [41] namely first Kaldi DNN implementation, second Kaldi DNN implementation and third Kaldi DNN implementation which are new models. These

DNN implementations are done in different ways [46] [47]. The second implementation of DNN is originally written based on the first one but it was extensively rewritten [41].The main difference between the first and second implementations is that in the first one it uses layer-wise pre-training based on RBM for training of the DNN whereas in the second one it does not use RBM directly, but instead of this it tries to apply the idea in another way. Furthermore, the first one uses early stopping using a validation dataset but the second one uses a fixed number of epochs and averages the parameters over the last few epochs of training [46]. The third DNN implementation is the newest of all and is an extension of the second one. Coming to which one to use, neither of them is more official than the others [48] so any one could freely use one of them that suits to him. Therefore, for this thesis the second one is chosen because it was found suitable in terms of computational resources.

The second Kaldi DNN implementation

This DNN implementation is also called Dan's DNN (Dan is the researcher and developer of the second DNN). It was originally written to support parallel training on multiple CPUs, although it has now been extended to support parallel GPU-based training. A single 5 core processor machine was enough for this thesis. This DNN has the following properties [41], which are common issues in implementing DNNs:

- Greedy layer-wise supervised training- Dan's DNN does not support Restricted Boltzmann Machine pre-training. Instead something similar to the greedy-wise supervised training is done. The network is initialized randomly with one hidden layer, trained for a short time (typically less than an epoch, meaning less than one full pass through the data), then remove the layer of weights that go to the softmax layer, add a new hidden layer and two sets of randomly initialized weights, and train again. This is repeated until the desired number of layers are obtained.
- Pre-conditioned Stochastic Gradient Descent (SGD)- Preconditioned SGD means SGD where, instead of using a single fixed learning rate, some arbitrary matrix-valued learning rate is used. This matrix is estimated per minibatch. The purpose of this matrix is to reduce the learning rate in dimensions where the derivatives have a high variance; this will tend to control instability and stop the parameters moving too fast in any one direction.

- Maximum change per minibatch- There is a limit on how much the parameters change per minibatch, which means the matrix representing the change in parameters of any given layer, on any given minibatch, cannot exceed this limit. If the change in a given minibatch exceed the maximum change allowed, the learning rate of that minibatch is multiplied by a constant less than one to make sure it does not exceed the limit.
- Learning rates- There are initial and final learning rates. These initial and final learning rates in a training setup must be specified by hand, and during training they are decreased exponentially, except for a few epochs at the end (typically for 5 epochs) during which they are kept constant. Half of the global learning rate, whatever it may be, is applied to the last and second-to-last layers of weights.
- Parameter shrinking and fixing- Shrinking means scaling the parameters of each layer by a separate factor, with the factors being determined by nonlinear optimization over a subset of the training data. This works better than using validation data [41]. It is called shrinking because it is expected that the resulting scaling factors will generally be less than one. Fixing is an operation to check whether neurons are over-saturated, and scales down the parameters for that neuron if so. Over-saturated is defined as the derivative at the nonlinearity, averaged over training data samples, being lower than a threshold [41].
- Minibatch size- Is a tunable parameter where a power of two number is used, typically 128, 256 or 512. Generally a larger minibatch size is more efficient because it interacts well with optimizations used in matrix multiplication code, particularly on GPUs, but if it is too large (and if the learning rate is too high), it can lead to instability in the update. So an optimum value should be determined through trial.

GMM based acoustic model

Kaldi has a class called AmDiagGmm which represents a collection of DiagGmm objects, indexed by pdf-ids that correspond to context-dependent HMM states. This class does not represent any HMM structure, but just a collection of GMM densities. The HMM structure, principally the topology and transition-modeling code and the code responsible for compiling decoding graphs, which provide a mapping between the HMM states and the pdf index of the acoustic model class, are represented by another separate classes, found not necessary to discuss it here.

Language Modelling, Decoding Graphs and Decoding

All of these tasks are supported by Kaldi. Language modelling is done using an external tool called, IRSTLM, integrated to Kaldi. Decoding graph is a certain combination of the acoustic model, pronunciation dictionary and language model done prior to decoding. Decoding is therefore done based on the graph. The toolkit uses a simple interface to hide the details of an acoustic model [40].

Speaker adaptation and feature transformation

Kaldi supports model-space adaptation using maximum likelihood linear regression (MLLR), feature-space adaptation using feature-space MLLR (fMLLR) also known as constrained MLLR [40] and speaker adaptive training (SAT). SAT is used to normalize speaker contribution, which means to reduce the speaker-specific variation and more accurately represent context-dependent variation and it is usually built from linear transform. The speaker-specific characteristics are captured by the MLLR. MLLR is to obtain the linear transformation which maximizes the likelihood of the adaptation data. In addition to this Kaldi supports the linear transforms, Linear Discriminant Analysis and Maximum Likelihood Linear Transformation (MLLT) usually used together (LDA+MLLT) to reduce the feature dimensions.

CHAPTER SIX

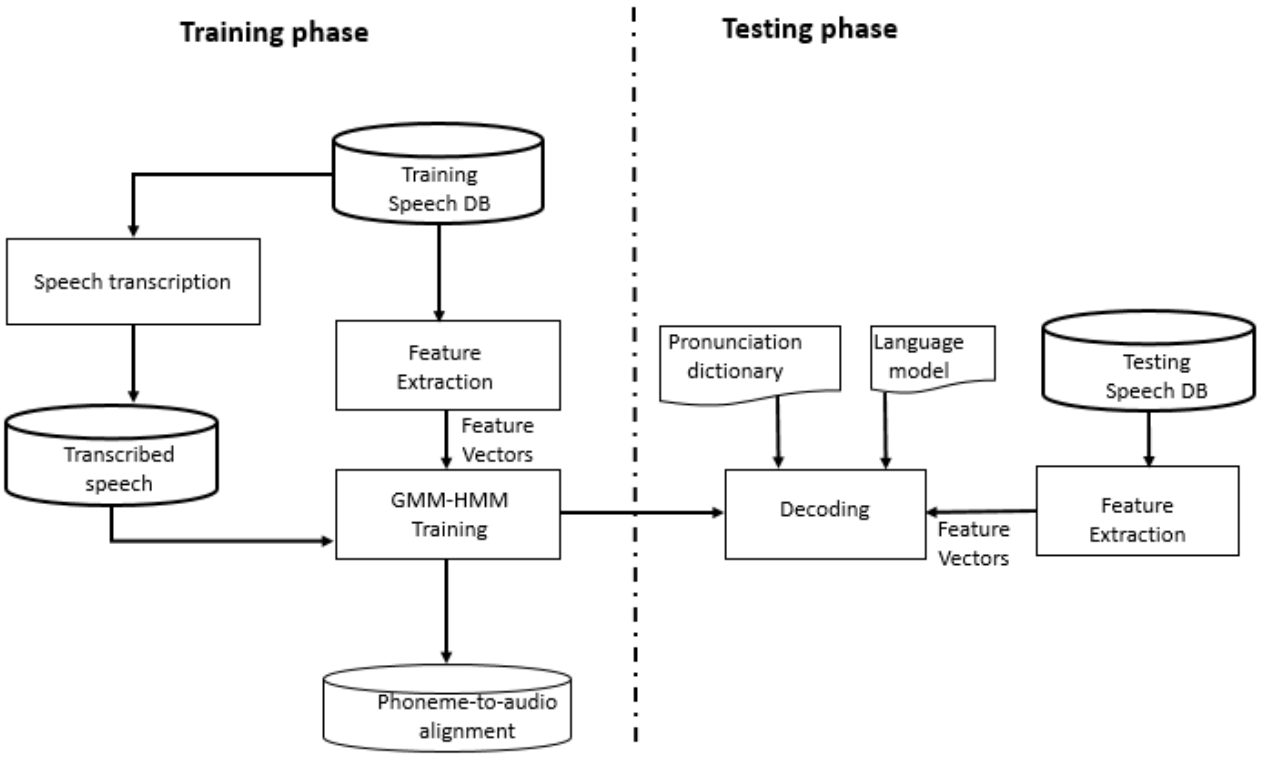
6. ANALYSIS, DESIGN AND IMPLEMENTATION

This chapter mainly focuses on three important parts of the thesis. Making an analysis of what a system should include and how the components of the system should be integrated, obviously results in a better system instead of starting to make the design directly. Similarly an implementation of a system should be done based on a design. This chapter therefore includes analysis, design and implementation of the system successively.

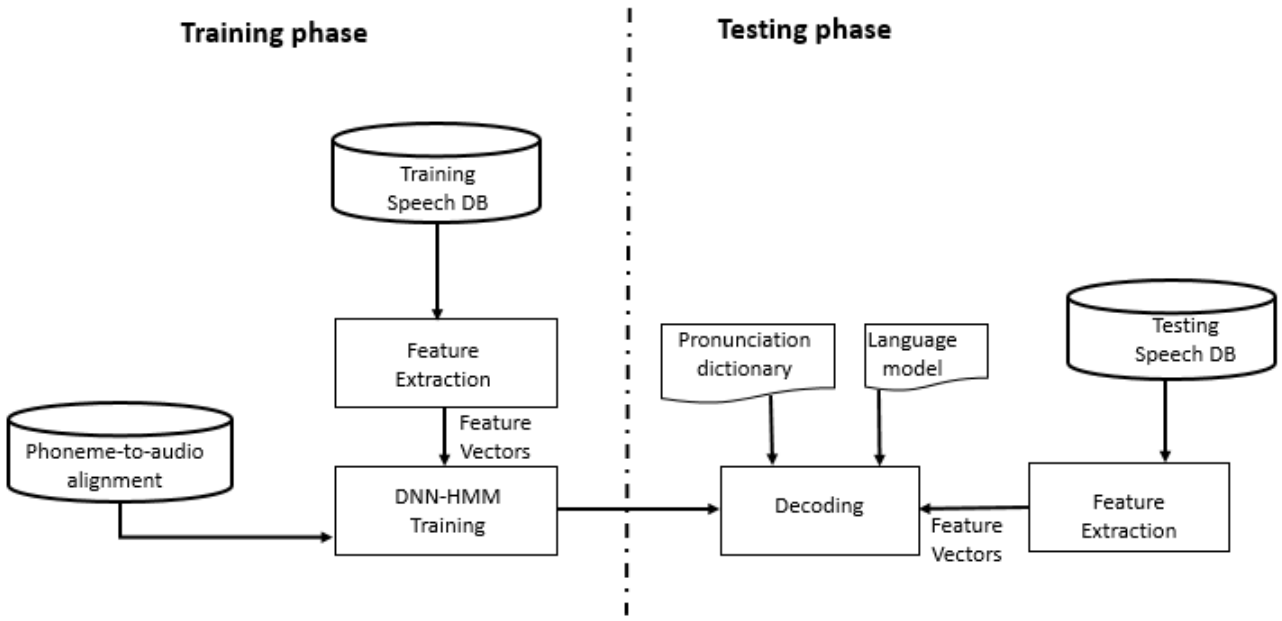
The components of a speech recognition system are grouped into two units: those that depend on a specific language and those that are common for all languages. The language dependent part is called Natural language processing (NLP) which is concerned with phonetic analysis and language modelling, and the second one is responsible for converting a given speech signal into the corresponding text by making use of an input from the NLP . So the difference in the NLP part makes each speech recognition system unique. Feature extraction, Acoustic modeling and Decoding are parts of the second unit. Generally, speech recognition system is the integration of these two units. Using these technical units, the whole process of the system is done in phases.

A complete speech recognition process is done in two phases: training and recognition (testing). By training, an acoustic model of the system is created. The acoustic modelling is done using two competing techniques in this thesis: GMM-HMM and DNNN-HMM. Features extracted using MFCC from the training speech data along with utterance-level transcription are used as the input for training of the GMM-HMM. Similarly the training of DNN-HMM uses the same input features as GMM-HMM, but it needs a phone-to-audio alignment found from the GMM-HMM training which is the target output used for the supervised training of the DNN. After the training is done, the system is tested for its level of recognition using a decoding module. The decoder needs a language model, a pronunciation dictionary and the acoustic model to do the recognition task. The language model supports for the acoustic model in providing the most probable sequence of words. As it is discussed, n-gram is the most commonly used technique for language modelling. For this research 1-gram was used due to the small amount of data, because 2-gram or 3-gram requires a lot more amount of data to properly train it. The pronunciation dictionary is the list of the words to be recognized with their sequence of phoneme combinations.

As it is stated in the problem statement, the focus of this thesis is on the acoustic modeling part to show if using DNN-HMM is better than GMM-HMM. So the design of the system includes both approaches. Most of the components of the system are the same for both approaches except at the point where they are applied, and a slight difference in the data used for training them. Figures 6-1 (a) and (b) show the architecture of these approaches.



a) Architecture of Tigrigna GMM-HMM system



b) Architecture of Tigrigna DNN-HMM system

Figure 6-1 Tigrigna speech recognition system architecture

6.1. Data Collection

As it is mentioned in the literature review, there is no open or commercially available Tigrigna speech corpus. So the corpus for this thesis is prepared by the researcher from scratch. The preparation of the data is done in two stages: first the text corpus was prepared and based on it the speech corpus was collected. The main difference in preparation is the source they are collected from. A text corpus is collected from written materials such as books, magazines, newspapers, and so on whereas a speech corpus can only be collected from speakers (humans) of the language of focus.

6.1.1. Text corpus preparation

Text corpus is a purposely selected and processed list of sentences or other smaller units like phrases or words. The preparation of a text corpus is the very beginning step in the process of speech recognition system development for languages, such as Tigrigna in this case, which do not have a commercially available corpus. The corpus for this thesis was prepared with as much care as possible by the researcher.

A total of 163 words have been selected for the text corpus. The words were mainly collected from a popular Tigrigna newspaper called “Weyn” and some were added by the researcher from his experience. The researcher has not used any formula to fix the

number of words, it was just thought that number is enough for the purpose of the research which is comparing the performances of GMM-HMM and DNN-HMM using fair amount of training and testing data. The words were selected in such a way that they are phonetically rich and balanced meaning the words are those that have as many phonemes as possible and that the frequency of the phonemes in the corpus be nearly equal. It has also been tried to make the words selected be the most frequently spoken words so that they can be used in future developments of similar systems. Abbreviations and contractions are avoided and numbers are written in textual form. This means all the words are written as they are pronounced by the speakers of the language. For example 3 was written as "ሰለስተ" and "መምህር" as "መምህር".

6.1.2. Speech corpus collection

Speech corpus is a spoken form of the text corpus, which was collected by the researcher for this thesis. Collecting a speech corpus is more challenging than text corpus due to many factors such as recording device, environmental noise, gender and age of speakers to mention the main factors. These factors were considered during the collection of the speech for this thesis. Generally, two speech types were necessary for this thesis: one collected from a silent environment that can be called a clean speech and the other from a noisy environment, a noisy speech.

The clean speech was prepared first. It has been tried to follow the same techniques and procedures during the entire collection time of this data. All the recording was done using a single Tecno L8 Lite smart phone on an office environment. A total of 86 speakers were participated, of them 66 are male and 20 are female, each speaker speaking all the words. In the choice of the number of male and female speakers it is clear that it should be balanced, even though this is not a problem for the objective of the work. The reason for the smaller number of female speakers here is due to the absence of the required number in the area were the speech was collected. During the recording, before starting to record, the speakers were told about the process to follow. The recorded speech was being played and it is recorded again if it does not satisfy the required quality. It has also been tried to include a variety of speakers as summarized in Table 6-1.

Gender	**Age**		**Recording environment**
	Range	Number of speakers	
Male(66)	15-18	16	Office
	20-30	40	“”
	40-60	10	“”
Female(20)	10-15	5	“”
	16-30	10	“”
	40-60	5	“”

Table 6-1 Speaker distribution for recording

To prepare the second type of speech, the procedure followed when preparing the clean speech has to be repeated except the environment has to be noisy. But this was found difficult in terms of resources. So it was prepared in another way. An artificial noise was generated and added to the clean speech. This was used as the second type of speech. The noisy signals are generated using the following mathematical way:

*Noisy signal=original signal+randn(size(original signal))*1/100,* (6.1)

where *randn* generates a random number as a function the original signal, which is actually the noise, divided by 100 to control the level of the noise.

6.2. Feature Extraction

Speech recognition system is one that involves a task of classification to classify an incoming audio signal to one of the classes. So the purpose of this feature extraction module is to extract the features that contain the necessary information to discriminate among the classes. The input to this module is the speech data and it gives out an appropriate feature vectors. As it is mentioned in the literature review, the best feature extraction technique for speech recognition is MFCC. The extraction process is made per frame and the dimension of the feature vector for each frame is 39. The 39-dimensional MFCC features are spliced in time taking a context size of 7 frames (i.e., ±

3), followed by dimensionality reduction to 40 using linear discriminant analysis (LDA) [43], which is the dimension used as input for the next stage.

6.3. Acoustic Modeling

In this module the model used for classification of the incoming frames is created. A classifying model is created through training using the training data and appropriate training algorithm. The types of training techniques and algorithms are dependent on the modeling approach being used. As it is already mentioned in the preceding chapters, this module is done using the two approaches: GMM-HMM and DNN-HMM where the purpose is to compare them by the effect they have on the whole system and take the better. Both have a common database of training speech, actually the features of the speech from the output of the feature extraction module. Their difference is, the first approach starts the training from scratch whereas DNN-HMM makes use of a support from GMM-HMM (takes a phoneme-to-audio alignment of GMM-HMM to guide its training), as it can be seen in Figure 6-1 (a) and (b).

6.3.1. GMM-HMM

This Hybrid approach is used to create a model of speech acoustics. The two probabilities in the HMM system, that is the transition probability and observation probabilities, are the tools of HMM for modelling of the acoustics. So in this hybrid structure, the observation probabilities are determined by the GMM. The transition probabilities and the number of GMMs are given reasonably guessed initial values and then they are modified through training until they have optimal values. The training data for this approach includes the features extracted from the training speech database and the utterance-level transcription of this speech database. Which means the training in this approach starts from scratch. After training the model, a trained acoustic model, which is ready to be used for classification, and an alignment of phonemes to the corresponding audios is produced. The optimum values of the parameters of the HMM are determined using the algorithms used for solving the three HMM problems, as discussed in chapter 5. As it is already mentioned in the literature review, a triphone modelling is a better approach for modelling of the phonemes. But according to the toolkit, the triphone model is developed starting from monophone training.

6.3.2. DNN-HMM

Like GMM, DNN is used to give the probabilities of the observations of the HMM states. Probabilities of the observations obtained from GMM are taken directly as provided, but unlike to that the probabilities obtained from the output of DNN are called posterior probabilities to indicate that there is a prior probability. Prior probability is the relative frequency of each HMM states. So when using DNN, the probability of the observations is obtained by dividing the posterior probability assigned to each HMM by its prior probability.

6.4. Decoding

After the system is sufficiently trained it is tested to determine how much the system is capable of recognizing new speech by using testing data from the testing speech data base that did not appear during the training. The purpose of the decoding module is therefore to do this testing (recognizing) process. It is the last component of the recognition process where it takes in a pronunciation dictionary, acoustic model and language model to give out the corresponding textual form of the spoken unit. For an incoming speech signal, the decoding module determines the best matching sequence of HMM states using the acoustic model, where each HMM state is a phone. It then determines a word from the pronunciation dictionary that could be formed by the combination of the sequence of the phones determined based on the acoustic model. But in some cases there may be two or more possible words pronounced the same way which definitely means they have the same sequence of phone combinations. Even though this situation is rare, it can create a confusion to isolated word recognition. But it is not a problem for a continuous speech recognition system. A 2 or more gram language model is used to determine the correct word for that phone sequence.

6.4.1. Pronunciation dictionary

It is the list of the words that the system has to recognize, each word followed by its phone combinations. It provides the words corresponding to the uncovered sequence of HMM states (which are the phones from the acoustic model) for the decoding module. That is the identified sequence of phones from the HMM model are tied up into one keeping their sequence and the dictionary is referred for the corresponding word of this tied sequence of phones. The preparation of this dictionary starts with a collection of a raw text. The

whole process of the preparation is done manually. The flow of the complete process is shown as in Figure 6-2.

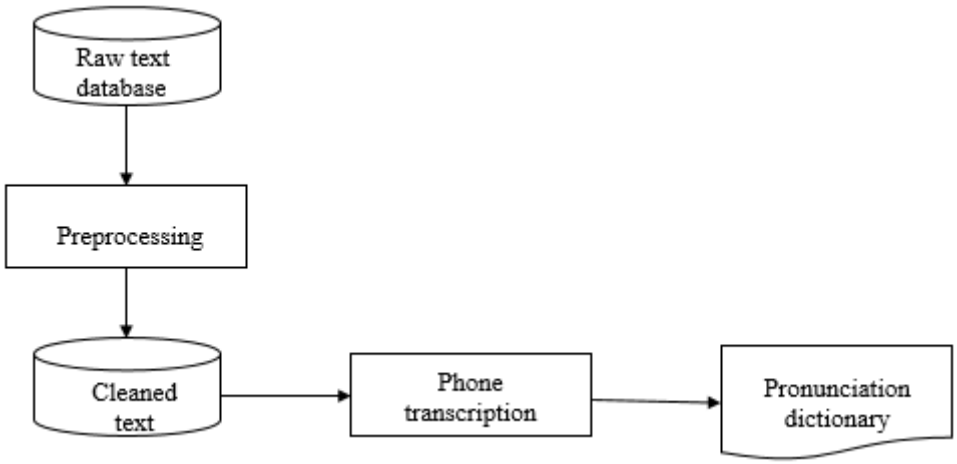


Figure 6-2 Preparation process of pronunciation dictionary

Preprocessing: after the raw text is collected as discussed in section 6.1.1, it is preprocessed to make it cleaned from inappropriate formats. In this step, abbreviations and contractions are avoided and numbers are written in textual form. This means all the words are written as they are pronounced by the speakers of the language. For example 3 was written as "ሰለስተ" and "ሙም/ር" as "ሙምህር".

Phone Transcription: is the process of writing the words in terms of the phones they are consisting of. A sample of the transcribed words can be shown in Appendix C. There is an international convention called International Phonetic Alphabet (IPA) for representing sounds of most languages including Tigrigna. But in this thesis, to comply with the standards of Kaldi, the sound representation of Tigrigna is modified to a suitable form as it can be shown in Appendix B.

6.5. Implementation

Except the data collection (text and speech) and preparation of the dictionary, which was done manually as discussed in the preceding sections, the rest of the components of the system are implemented by adapting the appropriate Kaldi tools. Kaldi has a certain file structure for preparing the training and testing data. A customized structure for this research is shown in Figure 6-3. The required data for this thesis was organized and prepared according to that structure and format. The language model, Acoustic Model

and Decoding modules are implemented by adapting the corresponding Kaldi commands and scripts. The commands are called from scripts, typically shell scripts for this work.

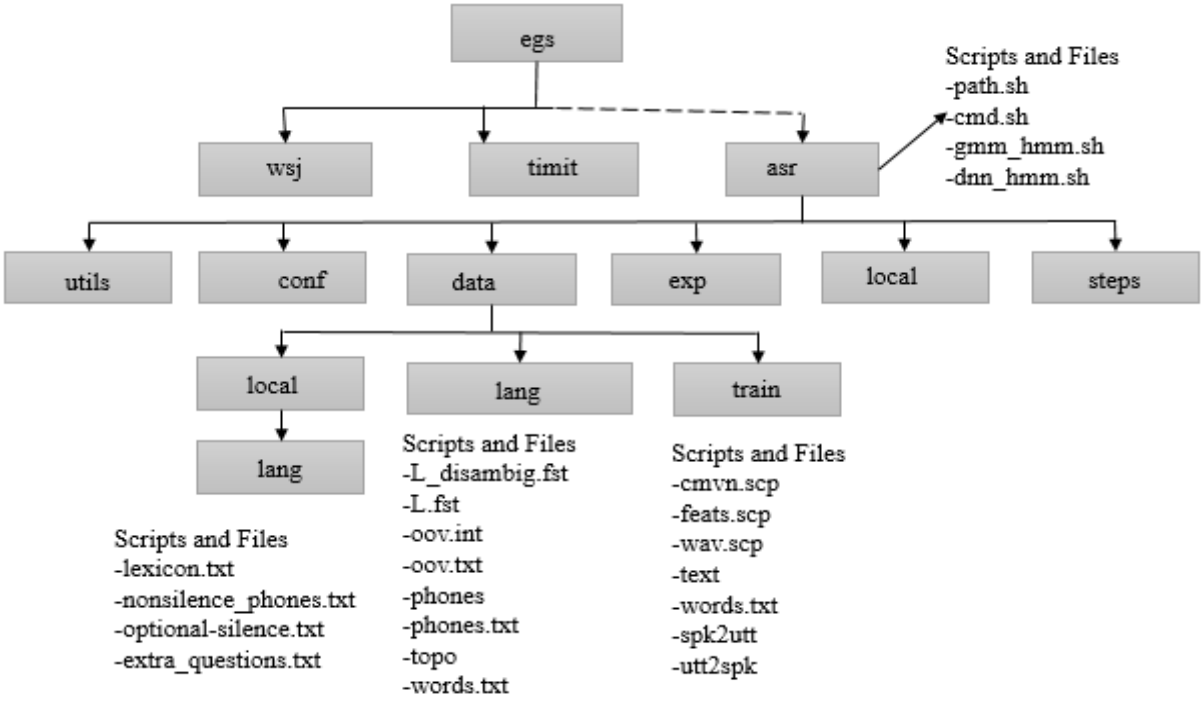


Figure 6-3 Kaldi file structure

The top-level Kaldi directories are egs, src, tools, misc and windows [4]. From these, the directories used for this work are egs and src where, egs stands for ‘examples’ and contains example training recipes for most major speech corpora and src stands for ‘source’ or ‘source code’ and contains most of the source code for programs that the training recipes call. Training recipes are available for the Wall Street Journal Corpus (wsj), TIMIT (timit), and many others. As can be seen from Figure 6-3 another directory named asr is shown from the figure which is a training recipe created for this thesis.

For each training recipe directory, there is a standard sub-directory structure. The top directory contains the run script (run.sh) which is replaced by gmm_hmm.sh and dnn_hmm.sh for the asr recipe, as well as two other required scripts (cmd.sh and path.sh). The sub-directories are conf (configuration), data, exp (experiments), local, steps, and utils (utilities). The directories primarily used for thesis are data, exp and conf. The data directory houses information relevant to the data such as transcripts, dictionaries, etc. The exp directory contains the output of the training and alignment scripts, or the acoustic

models. The files shown in the figure are only those that are needed for the development of the system and the conf directory has configuration files for setting values of some parameters. Files that are not needed are removed. Some of these files were prepared by writing a script from scratch and some by adapting Kaldi provided scripts. The files text, spk2gender, spk2utt and wav.scp were prepared by scripts written from scratch while the rest of the files were generated by adapted Kaldi scripts.

6.5.1. Creating the training files

a) Creating files for data/train and data/test

The files in both directories contain information regarding the specifics of the audio files, transcripts, and speakers. Both the directories use similar names for their files and the files are created the same way but different in content as the speech data they refer is different. The files are:

Spk2gender: this file tells about the speakers' gender. It is written according to the format given below:

Format: <speakerID> <gender>

where speakerID is a speaker's folder name.

text: this file contains every utterance matched with its text transcription. It is a text file with the format given as:

Format: <uterranceID> <text_transcription>

utteranceID was constructed by concatenating speakerID with the .wav file name without the ".wav" ending.

wav.scp: this file is a connection of every utterance with an audio file related to this utterance. It was prepared according to the format given as:

Format: <uterranceID> <full_path_to_audio_file>

utt2spk: this file tells the ASR system which utterance belongs to particular speaker. It has the following format:

Format: <uterranceID> <speakerID>

b) Creating files for data/local/dict

This directory contains language specific data based on the corpus at hand. For example, the lexicon (pronunciation dictionary) contains words and their pronunciations only for those that are present in the corpus. The files in this location are discussed as follows:

lexicon.txt: this file contains every word from the dictionary with its phone transcriptions. It is prepared according to the pattern given as:

Format: <word> <phone 1> <phone 2> …

Nonsilence_phones.txt: as the name indicates this file contains phones that are non-silence, which are present in the corpus.

Format: <phone>

silence_phones.txt: contains a 'SIL' (silence) and 'UNK' (unknown word) model. It is created using a simple script as:

echo –e 'SIL'\\n'UNK' > silence_phones.txt

optional_silence.txt: it simply contains a 'SIL' model. It can be created manually, or using a script as:

echo –e 'SIL'> silence_phones.txt

corpus.txt: this files contains the whole text corpus, but is in the data/local directory, which is one level higher than the directory under discussion.

c) Creating files for data/lang

All the files in this directory were created using the Kaldi script utils/prepare_lang.sh as:

utils/prepare_lang.sh data/local/dict "<UNK>" data/local/lang data/lang

After the script shown above is being executed, all the files shown under the data/lang directory are created.

d) Creating files for conf

Different configuration files are put in this directory. A single configuration file, mfcc.conf, was used for this work. It is a simple file and was prepared by hand as shown below:

*--sample-frequency=1600. T*his is to inform the feature extractor module that, the speech was recorded at a 16KHZ frequency.

6.5.2. Feature Extraction

The speech features were extracted using Kaldi script steps/make_mfcc.sh. This script actually uses a program called *compute-mfcc-feats,* to extract the features. This program looks for non-default parameters in mfcc.conf file in the conf folder, getting only one for this work's case which is the sampling frequency. The parameters found in this configuration file are passed to the corresponding binary files. Finally, the script generates two output files: feats.scp and feats.ark. The first one is a mapping of the utterance ids to positions in the second file which is the actual data (the extracted features). The process is done to extract the features from both the training and test data to get the corresponding training and testing features.

6.5.3. GMM-HMM training

This model can be created using either monophone or triphone training, the second being the better approach as already discussed. In this thesis both of them are used instead of using only one for a purpose. The monophone was used as a base for triphone. Both of these trainings were done using adapted Kaldi scripts. Each of the training script takes a similar argument structure with optional arguments preceding those. The one exception is the first monophone training pass. Since a model does not yet exist, there is no source directory specifically for the model. The required arguments are always:

- Location of the acoustic data: data/train
- Location of the lexicon: data/lang
- Source directory for a base model: exp/lastmodel
- Destination directory for the trained model: exp/currentmodel

All the training scripts for GMM-HMM, as used for this work, can be shown in Appendix E.

a) Monophone training and alignment

They are the first part of the training procedure and are mainly used to bootstrap training for later stages. The training for this approach is done using customized Kaldi script.

b) Triphone training and alignment

Triphone is where a single phone is represented by three HMM states making the number of states and parameters very large, which is difficult to manage. So to make this manageable decision tree is used. For training the triphone model it needs to specify the number of leaves or HMM states on the decision tree, and the number of Gaussians. These numbers should be more than what may be actually needed. To elaborate what it means, assume there are 29 phonemes in the lexicon. Since each phone is being represented using 3 HMM states, it brings the number to a minimum of 87 HMM states. For example, with 2000 HMM states, the model has a flexibility to decide if it may be better to allocate a unique HMM state to more refined allophones of the original phone. The exact number of leaves and Gaussians is decided from the training. The number of Gaussians is made based on a constraint that, it must be always greater than the number of leafs to have some reserves for future in case the number of states may increase.

6.5.4. DNN-HMM training

When training DNN-HMM, most of the training is done on the DNN part because the HMM is taken from the trained GMM-HMM model. The only thing modified for the HMM in this training is the observation probabilities, which are obtained from the output of the trained DNN.

The toolkit's implementation (Kaldi DNN2) has been customized in some way. The original implementation was done considering multiple machines to run it. Additionally these machines are assumed to have GPUs. So in this work the code was modified to run on a single machine using CPU. Having implementing the logic of the DNN, its parameters are left to be tuned by the researcher. Tuning the parameters appropriately is a very determining factor for the performance of the system. So, the optimum values of the important parameters for this system has been determined through many trials of training. Typically, the tunable parameters are the number of hidden layers, the dimension of each hidden layer, minibatch size, hidden layer addition interval and number of epochs. Default values were taken for the other parameters that had less effect on the system's performance.

The parameters were tuned in such a way that one is selected at a time and others are kept constant. The value of the selected parameter is modified until an optimum value, which gives the best result, is found. This process is applied for all the tunable parameters. The

process of determining an optimum value for all the parameters is given in the flow chart below.

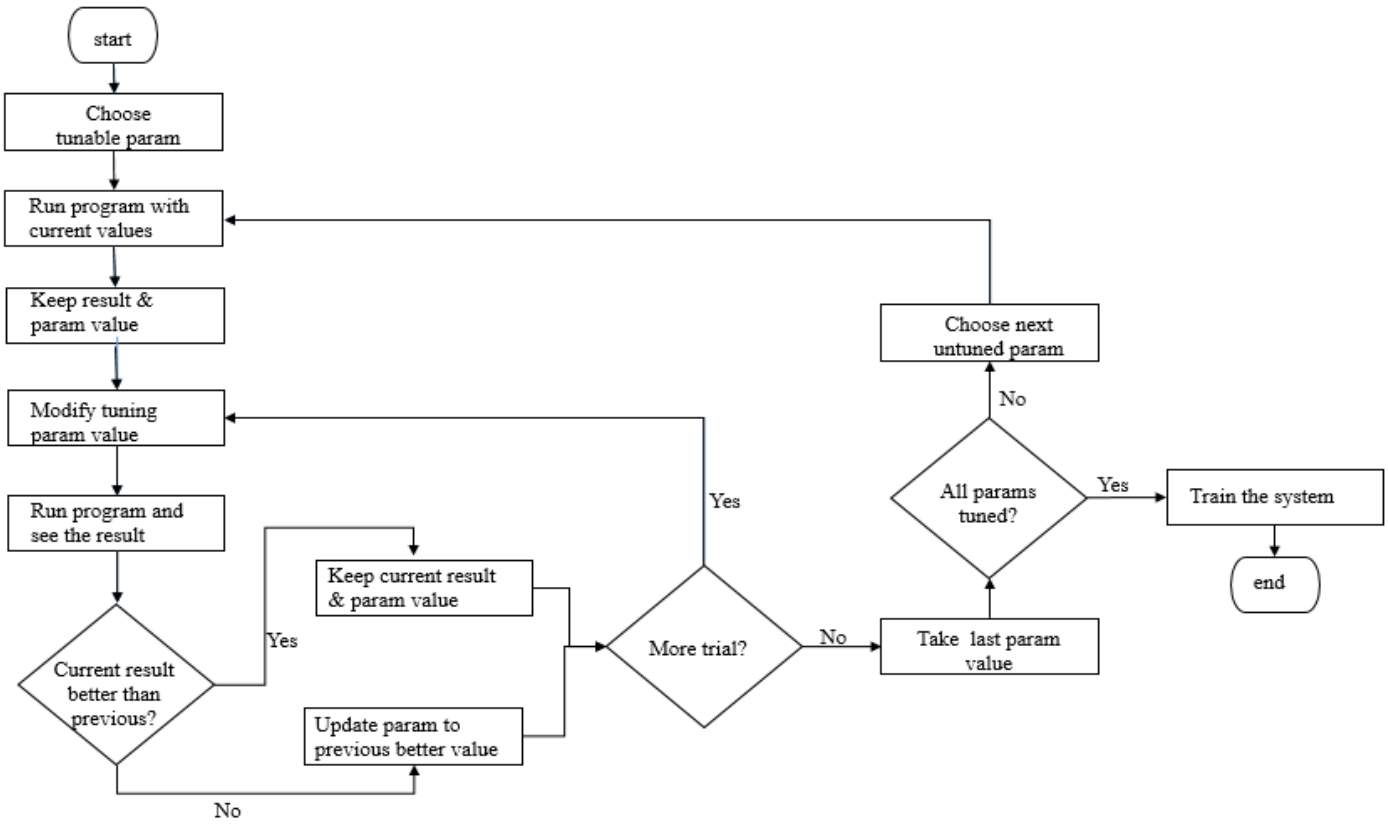


Figure 6-4 Process of DNN parameters tuning

6.5.5. Decoding

Decoding is done using the trained acoustic model, the pronunciation dictionary and the language model (even though less important for this work). But it is not made directly on the components described. A decoding graph which is a combination of the listed components is first created and then the decoding is performed based on this graph. Both the creation of the graph and decoding are done using the toolkit. Even though it is the last step in a speech recognition process, the result of the decoding part is given to another analysis module for interpretation to report the result in a human understandable form.

6.5.6. Evaluation

The evaluation of the system was done using the Kaldi analysis tool called scoring. This module analyzes and calculates the accuracy of the system. The performance of the system was evaluated using two of the most common measures, accuracy and error,

which are commonly given in percentage notation. These measures can be calculated on phone and word level. The second one, named as word error rate (WER), is used for this work. The calculation of the error is done based on the following elements [27].

- Substitution: At the position of a unit (word or phoneme), a different unit has been recognized.
- Deletion: In the recognition result a unit is omitted.
- Insertion: The result contains new units than were not spoken

The measures are therefore given as follows:

$$\text{Accuracy}=\left(\frac{\text{N}-\text{D}-\text{S}-\text{I}}{N}\right)*100\%$$

$$\text{WER}=\left(\frac{\text{D}+\text{S}+\text{I}}{N}\right)*100\% = 100\% - Accuracy$$

Where D is deletion, S is substitution, I is insertion and N is the total number of words.

Based on this, the results (in terms of WER) for the different approaches are shown and discussed in the next chapter.

CHAPTER SEVEN

7. RESULTS AND DISCUSSION

This chapter presents and discusses the results found from the works and experimentations of this thesis. The very first task for the experimentation of the system is preparation of an appropriate data. So the necessary dataset, text and speech data, are prepared accordingly. For testing of the systems' robustness, an artificial noise was added to a copy of the original speech data, resulting to have two speech datasets. Having these datasets, the experimentation environment was setup; the values of important parameters of the system were selected through training, and the research questions and problems are then discussed according to the results of the experiments.

7.1. Text and Speech data

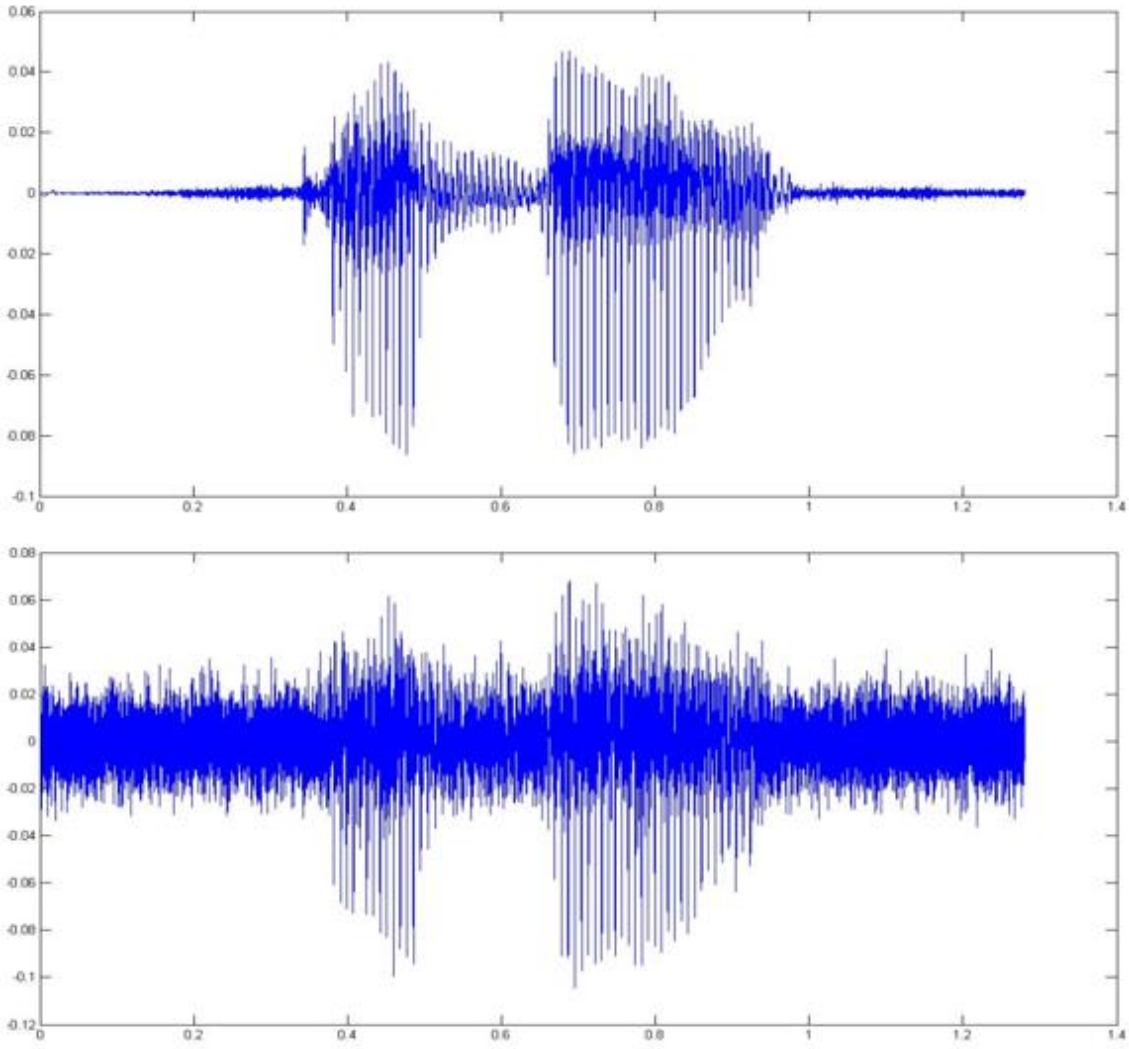
The text data has been prepared based on a predesigned approach. That design is implemented manually by the researcher. Part of the result of this work is shown in table 7-1. Typically, this data is the pronunciation dictionary.

Word	**Transcription**
Babnnet	B ab n et
Cew	C ew
Derejja	d er ej ja
Glgallot	g l g al lo t
Hbret	H b r et
Jnqfat	J n q f at
meCerexxa	m eC er ex xa
Tmhrti	t m h r t iy

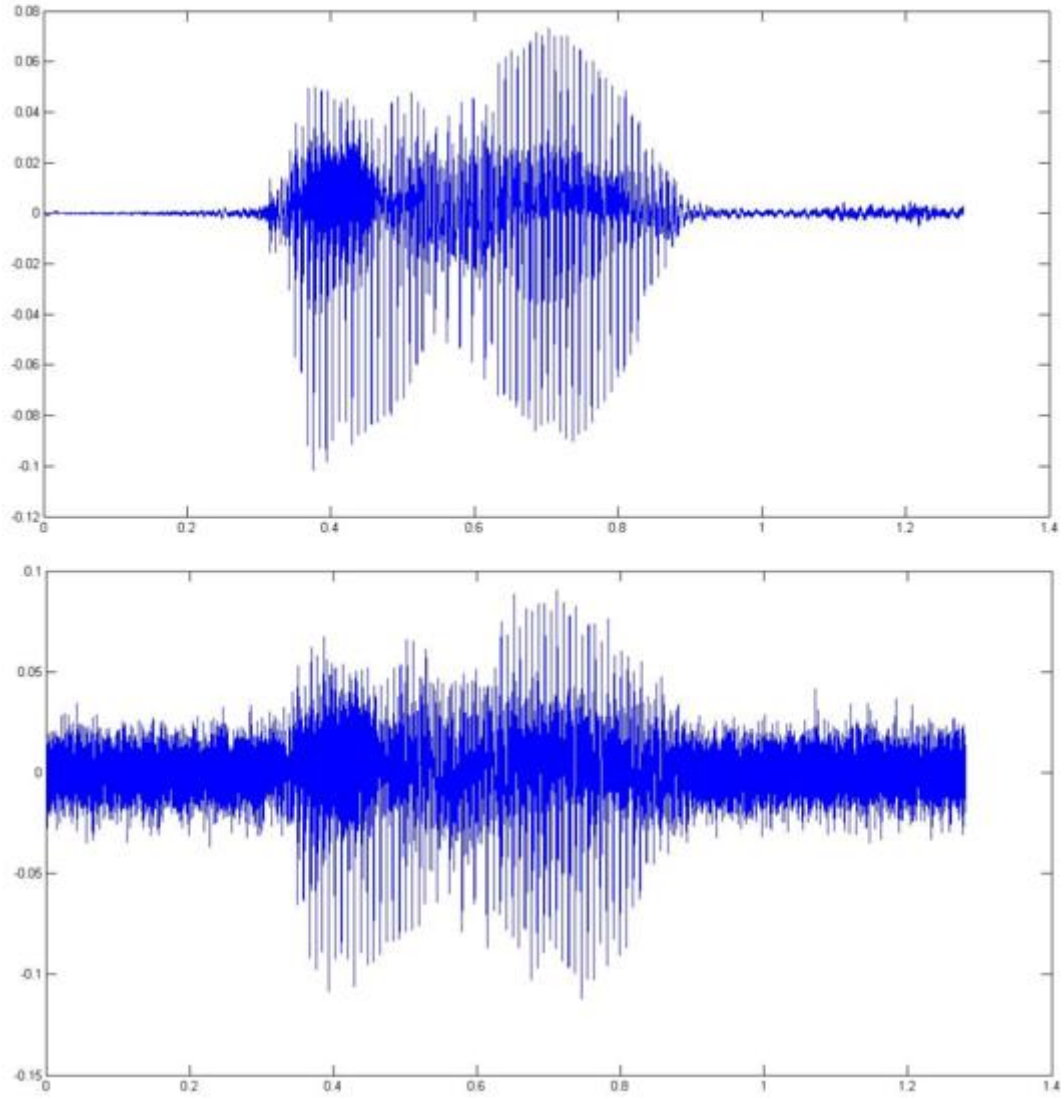
Table 7-1: A sample of pronunciation dictionary

As it can be seen from the above table, the first column is for the words and in the second column are the phones that make up the corresponding words. As it can be shown from this table, the Tigrigna words are written using English alphabets and similarly the phones are written in terms of English alphabets instead of using the conventional IPA representation. This is intentionally done for convenience and to comply with the Kaldi format. Following the preparation of the text data, the collection of the speech corpus was done. The collection of the original speech data was already discussed in the previous

chapter. The noisy speech was prepared by adding an artificially generated noise to the original speech according to equation (6.1). A graph for a sample of the original and the noisy speeches can be shown in Figure 7-1.



a) Speech sample1



b) Speech sampl2

Figure 7-1 Samples of clean and corresponding noisy speech signals

Figure 7-1 (a) consists of the original clean speech (the first one) and the corresponding noisy speech. Similarly (b) is a plot of another speech signal (clean and noisy). As can be shown from the results of the experiment the noises has come to reduce the performances of both approaches. But the reduction level is different in the approaches. The performance of GMM-HMM is reduced more than DNN-HMM. This means that this noise has made to compare the noise robustness level of the systems.

7.2. Experimental setup

The setup of the experimentation environment requires setting the optimum values of the training parameters. The parameters for GMM-HMM are determined by the toolkit itself through training, except the initial values are given by the researcher such that the initial GMM value was set to 2500 and 2000 initial HMM states whereas DNN parameters are determined by the researcher through many trials. To select values for these parameters, different values were tried until one with a best performance is found. The most important parameters are number of hidden layers, dimension of each hidden layer, minibatch size, hidden layer additions per iteration and number of epochs. When tuning these parameters, different values were tried such that the number of hidden layers were varied from 1 to 4, the dimension of each hidden layer 64 to 400, minibatch size from 64 to 512, 2 hidden layer additions per iteration and number of epochs from 5 to 20. As it is discussed in chapter 6, all the DNN parameters have been tuned the same way. Two representative parameters, the number of neurons in each layer and the number of hidden layers, were chosen to discuss the result of the optimization process for the whole parameters. These are the last two parameters tuned in the order listed. So the performance of the system found with the best configuration of the last parameter tuned is the value taken as the best result and compared with the best result of GMM-HMM.

The figures below show the tuning of the two DNN parameters. The first one is for the number of neurons in each hidden layer and the second one is for the number of hidden layers. As it can be seen from the second one of figure 7-2, the tuning of the number of hidden layers starts from 2 and is incremented by 1 until the performance starts to decrease.

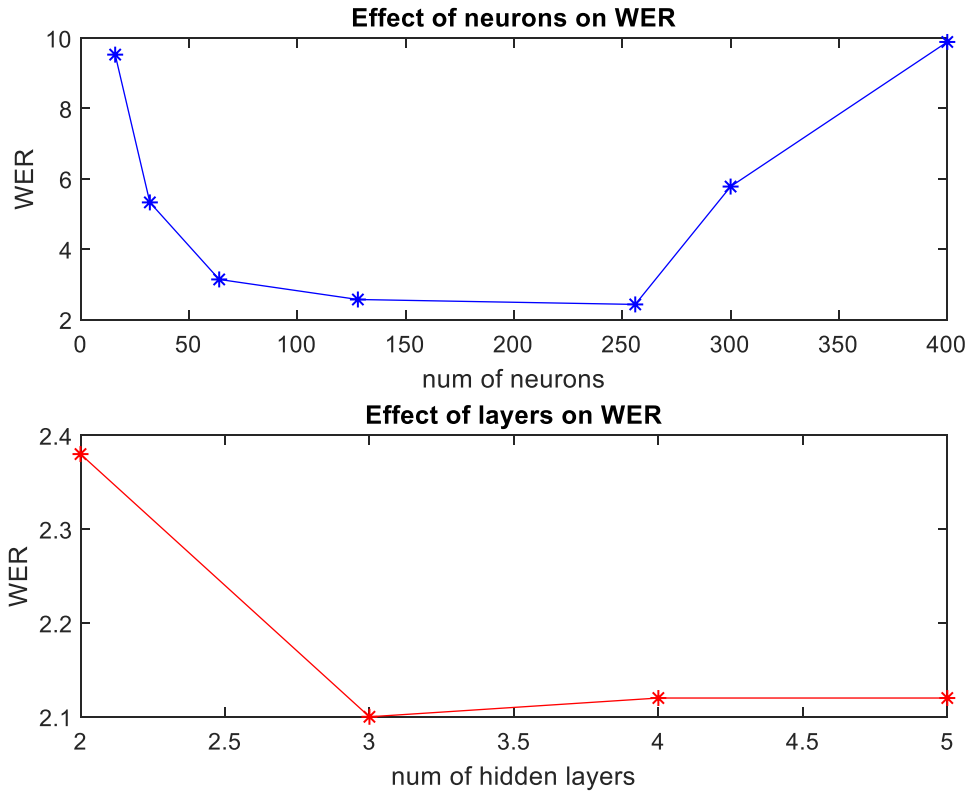


Figure 7-2 Sample of DNN parameters tuning

From that graph, the WER found from small number of hidden layers is greater. As the number of hidden layers is increasing the WER is decreasing up to some point from which it starts to increase again. The point at which the graph starts to go up is where an over fitting is starting to happen. This is indicating that there is a number that specifically suits for the system and that is the optimum value which is 3 from figure 7-2. Applying the same procedure for the other parameters, the optimum values are: minibatch size of 128, 20 epochs, 256 neurons per hidden layer and 2 iterations per hidden layer addition. This configuration has resulted in an improved performance of DNN-HMM system as can be shown from Figure 7-2. This optimization may not be the best possible one, it is just taken because an improved performance was found. Tuning the parameters further could result in a more improved performance of the DNN-HMM.

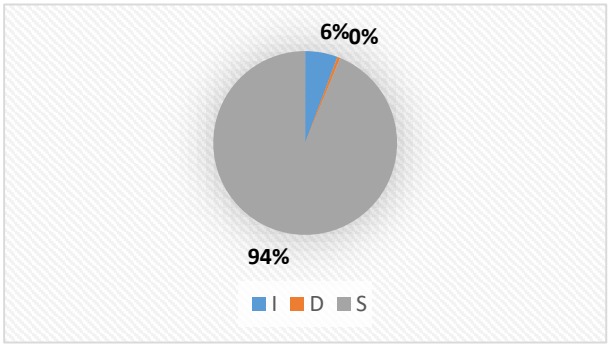
After the optimum values for the parameters are set, the final architecture has come to have the following setup for the system experimentation: the dimensions of the input and output layers are fixed that are determined from the GMM-HMM. The input layer has 40 neurons which is equal to the dimension of the input feature vectors, has an output layer

dimension of 1448 neurons which is equal with the optimum number of HMM states, 3 hidden layers, 256 neurons per each hidden layer, minibatch size of 128, 20 epochs and 2 iterations per hidden layer addition which is used to simulate the greedy layer-wise training. This configuration is applied on the adapted Kaldi DNN script. The script can be shown in Appendix E.

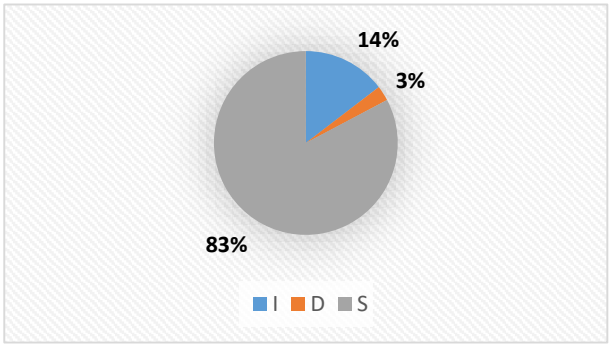
7.3. System Experimentation and Answers to Research Questions

7.3.1. The evaluation metrics

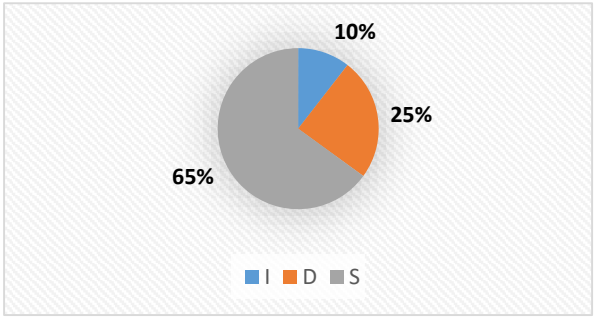
Using the pronunciation dictionary along with the trained acoustic model, is used as an input for the decoding module. The output of the decoding module is not directly understandable by humans. So it is converted to a meaningful output by a score making module. This module uses the number of insertions (I), number of deletions (D), number of substitutions (S) and the total number of correct words for making an analysis and evaluation on the decoder's output. The result of the scoring is shown in table 7-2. The table presents the results from the different methods. Clearly, we can see different values of I, D and S for each of the methods from the table. In all of the methods the substitution error has a highest value that means it is contributing most to the overall recognition error. From this, we can take a note that working to reduce the substitution error can reduce the recognition error more. Substitution error is where a word in the utterance is transcribed as a different word. This mostly happens due to the effect of the environment and nature of the speaker when recording the speech. Or there may be some people who are not able to pronounce some words correctly. For example, the word mat might be misinterpreted as bat due to these reasons. The I, D and S error compositions can be shown clearly in figure 7-1 from the different methods.



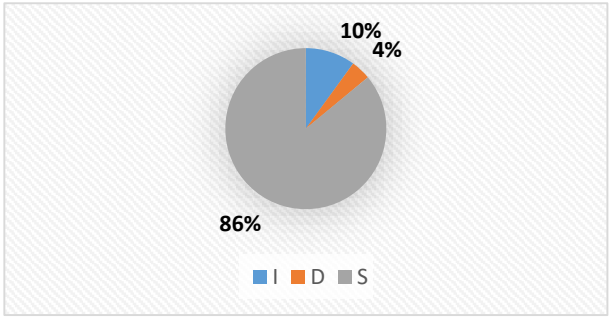
a) I, D, S in Monophone



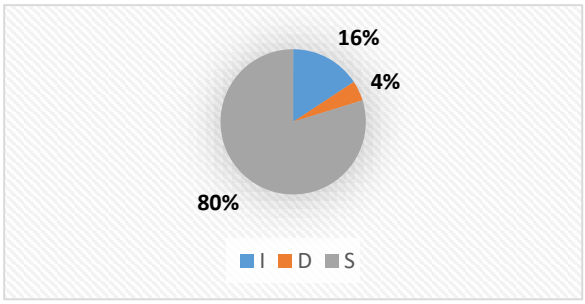
b) I, D, S in delta+delta-delta



c) I,D,S in LDA+MLLT



d) I,D,S in LDA+MLLT+SAT



e) I,D,S in DNN-HMM

Figure 7-3 I, D, S percentage errors in the total recognition error

7.3.2. System results

Acoustic modeling method	**N**	**I**	**D**	**S**	**WER (%)**
Monophone	4238	22	2	361	9.08
Δ+Δ-Δ triphone	4238	33	6	187	5.33
LDA+MLLT triphone	4238	21	49	130	4.72
LDA+MLLT+SAT triphone	4238	10	4	86	2.36
DNN-HMM	**4238**	**14**	**4**	**71**	**2.10**

Table 7-2 clean speech recognition results from the different modeling techniques

Model	Monophone	Δ+Δ-Δ	LDA+MLLT	LDA+MLLT+SAT	**DNN-HMM**
WER (%)	51.51	44.76	45.85	30.60	**24.54**

Table 7-3 noisy speech recognition results from the different modeling techniques

7.3.3. Research questions

a) **Research question 1: "***Can DNN-HMM perform better compared to GMM-HMM in modeling Tigrigna speech acoustics?"*

Table 7-2 is a result of the experimentation done using the different acoustic modeling approaches of the system on the clean data. Generally saying the system is done using the two competing approaches: GMM-HMM and DNN-HMM. As can be shown from the table there are more than one alternative approaches to implement GMM-HMM system. They are broadly two: the monophone and the triphone. But as can be seen from the table, there are more than one triphone based results. Each of the results are found from the different input processing techniques, already indicated there. The monophone system is the base for all the other implementations. Δ+Δ-Δ triphone is done based on monophone, LDA+MLLT triphone based on Δ+Δ-Δ triphone and LDA+MLLT+SAT triphone based on LDA+MLLT triphone. The Δ+Δ-Δ triphone is a context dependent approach but monophone is context independent, as it is widely discussed in the literature review. From this result, triphone has shown a significant performance improvement over the monophone. The next type, LDA+MLLT is a transformation of the input features dimension to that suitable for training. This has indeed reduced the error and result in a further improved system. SAT is the last performance improvement technique that Kaldi provided. It is used for reducing the variations in the input happened due to the varied nature of the speakers. Thus, it has reduced the error significantly. The result found from the SAT is the best result of all the possible GMM-HMM implementations. This result was then used as the baseline for the training of the DNN-HMM system. Therefore, as it can be inferred from the table, DNN-HMM has come with a better performance compared with the best result of GMM-HMM. Therefore the first research question is get answered. DNN-HMM has a better modeling power than GMM-HMM. Now, coming to the second research question, that is to test the level of noise robustness of DNN and GMM, the

experimentation was done with the noisy test data on the already trained models using the clean training speech data.

b) Research question 2: ***"****Is DNN based Tigrigna speech recognition system robust to noise compared with GMM based?"*

After testing the models on the noisy speech the result shown in table 7-3 is found. It can be observed that all the results in this table are a lot greater than the corresponding results in table 7-2. I another words, this result has got worse. This is natural to expect because in the second one a noisy speech was used that makes the recognition of the words challenging. That is, a system capable of filtering the words out of the noise is required, which is a noise-robust system. We can also see that the error from LDA+MLLT has get increased from the Δ+Δ-Δ that should normally be the reverse. This might happened due to the transformation of the noise which is really irrelevant that when the variation is reduced by SAT the result becomes better. Coming to the main point of the discussion, when the two trained models (GMM-HMM and DNN-HMM) are tested using the noisy data a significant recognition difference is found, taking the best results from both. That is DNN is found to resist for the noise better than GMM do.

7.4. Comparison with Previous works

As it was stated in the problem statement and already mentioned in the literature review, one related work is done for Tigrigna ASR. That system is a continuous speech recognition system, which is better than this thesis's work in terms of the goal of ASR systems. But the goal of this thesis's work is to investigate a better AM modeling technique. The AM part in that previous system was done using the GMM-HMM. In this thesis work another AM modeling technique called DNN-HMM is proposed. So to find out which one is better, both the methods are examined in this new system. Thus, the results found indicated that a DNN-HMM has a better modeling power, as it is already discussed above, and therefore it can be concluded that, the previous system could be improved by using DNN instead of GMM.

CHAPTER EIGHT

8. CONCLUSION AND FUTURE WORKS

This chapter summarizes the work done in this thesis and it points out the main findings. Finally it gives an idea of what should be done in future works.

8.1. Conclusion

In this thesis DNN-HMM is proposed for the acoustic modelling part of Tigrigna Speech Recognition System, to improve the performance over the previous similar system done based on GMM-HMM. The dataset is prepared by the researcher since there is no publicly available one so far.

The preparation of the dataset was one of the challenging tasks of this thesis. It is consisted of text data and speech data. The text data is collected from purposely selected Tigrigna newspaper and from the researcher's experience. The speech data is collected from a total of 86 different speakers. The speakers are selected to be from different age ranges and from both genders. All the speech was recorded in a silent environment using the same recording tool. Then extraction of the necessary features is done to prepare a processed and suitable data for the next modules. The extraction of the features is done using MFCC as it is found to be more appropriate, from the review of the literatures, for speech recognition systems than other related techniques.

Two separate systems, due to the method used for the AM part, are implemented and trained; the first one being GMM-HMM based and the second one using DNN-HMM. The implementation of both systems is done by the help of Kaldi, the speech recognition tool. The training of GMM-HMM is done first to use the phone-to-audio alignment it produced for the next system. The values of the GMM and HMM parameters are initialized to 2500 Gaussians and 2000 HMM states manually and modified to an optimum value through training using the toolkit. Having the trained GMM-HMM based system the training of the next one, DNN-HMM, is done using the alignment found from the training of the previous system. The DNN training parameters are experimentally set to optimum values such that a total of 3 hidden layers, 256 neuron per hidden layer and minibatch size of 128 are used. In addition to that, 2 iterations per hidden layer addition is used to simulate the greedy layer-wise training.

After the training of both systems, they are evaluated for their performance. Both systems are tested on the clean and noisy datasets. The purpose of the clean dataset is to test which one of the methods is a better performer on a normal speech, and the noisy data is used for testing robustness of the systems. So after making these evaluations it we found that, the best result of DNN-HMM is a WER of 2.10% on the clean data and 24.54% on the noisy data. Whereas using GMM-HMM the best result found is a WER of 2.36% on the clean data and 30.60% on the noisy data. This shows that DNN-HMM performs better than GMM-HMM and it is more robust in the case of noisy environment.

8.2. Future Works

In this work it was attempted to use a DNN-HMM based Tigrigna speech recognition system for an isolated words only. The following points are recommended for future systems:

1. Future works should extend this to a continuous speech recognizer
2. The scope of the research was to test the recognition performance of DNN-HMM system compared with GMM-HMM based system. A huge data was not necessarily required for this purpose. But to develop really applicable systems a sufficiently huge training and testing data should be used.
3. The speech data was collected from limited places which did not include all the possible dialects. Future systems should work to include all the possible Tigrigna dialects.
4. The pronunciation dictionary was prepared based on the knowledge of the researcher from his experience. But it is recommended to use the support of Tigrigna language experts.

REFERENCES

[1] R. D. Shaikh Naziya, "Speech Recognition System-A Review," *IOSR Journal of Computer Engineering (IOSR-JCE),* vol. 18, no. 4, pp. 01-09, 2016.

[2] B.H. Juang, Lawrence R. Rabiner, "Automatic Speech Recognition- A Brief History of the Technology Development", unpublished

[3] Katri Leino,"Breakthroughs in Automatic Speech Recognition Technology", July 2015

[4] W.Zielinski ,"KALDI FOR DUMMIES", unpublished

[5] Lawrence R.Rabiner,"A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", Proc.IEEE vol. 77, no.2, Feb.1989

[6] G. Hinton *et al*, "Deep Neural Networks for Acoustic Modeling in Speech Recognition", IEEE Signal Processing Magazine, Nov.2012

[7] Hafte Abera Miruth,"Hidden Markov Model based Tigrigna Speech Recognition", M.Sc. thesis, Department of Information Science, Addis Ababa University, Addis Ababa, 2009

[8] Shanthi Therese S., Chelpa Lingam, "Review of Automatic Speech Recognition Techniques", International Journal of Scientific Engineering and Technology, vol.2, pp.479-484,June 2013

[9] Preeti Saini, Parneet Kaur,"Automatic Speech Recognition- A Review", International Journal of Engineering Trends and Technology, vol.4, pp.132-136, 2013

[10] *LINGUISTIC ANALYSIS,* Langtech Corporation, Milford, Michigan, 2003

[11] Girmay Berhane ,"The Phonology of Tigrigna",M.Sc. thesis, Department of Master of Arts in Linguistics,Addis Ababa University,Addis Ababa,1983

[12] Wikipedia. (2017, April 26). *Tigrinya_language.* [Online]. Available: https://en.wikipedia.org/wiki/

[13] *Speech and Language Processing,* Prentice Hall, Englewood Cliffs, New Jersey, 2000

[14] Lawrence Rabiner, "Approaches to Automatic Speech Recognition by Machine", in *FUNDAMENTALS OF SPEECH RECOGNITION,* New Jersey, PTR Prentice Hall Inc., 1993, pp.45-65

[15] Kai-Fu Lee, "Finding Good Unit of Speech", in *Automatic Speech Recognition: The Development of the SPHINIX system*, 1st edition Carnegie Melon University, Springer Science+Business Media, 1989, pp.91-95

[16] Dhavale Dhanashri and S.B. Dhonde, "Isolated Word Speech Recognition System using Deep Neural networks", unpublished

[17] Namrata Dave, "Feature Extraction Methods LPC, PLP and MFCC In Speech Recognition",International Journal for Advance Research in Engineering and Technology,vol.1, pp.1-5, July 2013

[18] Urmila Shrawankar, "Techniques For Feature Extraction in Speech Recognition System- A Comparative Study", unpublished

[19] Shreya Narang, Ms. Divya Gupta, "Speech Feature Extraction Techniques- A Review", International Journal of Computer Science and Mobile Computing, vol.4, pp.107-114, March 2015

[20] Wiqas Ghai and Navdeep Singh, "Literature Review on Automatic Speech Recognition", International Journal of Computer Applications,vol.41, pp.42-50, March 2012

[21] Dong Y. and Li Deng, "Deep Neural Network- Hidden Markov Model Hybrid Systems", in *Automatic Speech Recognition: A Deep Learning Approach*, Springer-Verlag, London, 2015, pp.99-115

[22] Andre L.Mass *et al*, "Building DNN acoustic Models for large vocabulary speech recognition", Computer Speech and Language, vol.41, pp.195-213, June 2016

[23] Kishori R. Ghule and R. R. Deshmukh,"Feature Extraction Techniques for Speech Recognition: A Review", International Journal of Scientific and Engineering Research, vol.6, pp.143-147, May 2015

[24] Michelle Gutajar *et al*, "Comparative Study of Automatic Speech Recognition Systems", IET Signal Processing, vol.7, pp.25-46, Jan.2013

[25] Cristina Espana-Bonet and Jose A. R. Fonollosa,"Automatic Speech Recognition with Deep Neural Networks for Impaired Speech", unpublished

[26] Mostafa Shahin *et al*, "A Comparison of GMM-HMM and DNN-HMM Based Pronunciation Verification Technique for Use in the Assessment of Childhood Apraxia of Speech", INTERSPEECH, pp.1583-1586, Sep. 2014

[27] R.E. Grhun *et al*, "Statistical Pronunciation Modeling for Non-Native Speech Processing", Signals and Communication Technology, pp.6-17, 2011

[28] Febe de Wet *et al*, "Speech recognition for under-resourced languages- Data sharing in hidden Markov model systems", South African Journal of Science, vol.113, pp.1-9, Jan.2017

[29] M.A. Anusuya and S.K. Katti,"Front-end Analysis of Speech Recognition", International Journal of Speech Technology, vol.14, pp.99-145, 2011

[30] Solomon Berhanu ," Isolated Amharic Consonant-Vowel (CV) Syllable Recognition: An experiment using the Hidden Markov Model", M.Sc. thesis, Department of Addis Ababa university, Addis Ababa, 2001

[31] Kinfe Tadesse, "Sub-Word Based Amharic Word Recognition: An Experiment Using Hidden Markov Model", M.Sc. thesis, School of Information Studies for Africa, Addis Ababa university ,Addis Ababa, 2001

[32] Zegaye Seifu," Hidden Markov Model Based Large Vocabulary, Speaker Independent, Continuous Amharic Speech Recognition", M.Sc. thesis, Department of Informatics, Addis Ababa university, Addis Ababa, 2003

[33] Hussien Seid and *Bj¨orn Gamb¨ack,"* A Speaker Independent Continuous Speech Recognizer for Amharic*",* INTERSPEECH, pp.3347-3352, Sep. 2005

[34] Solomon Teferra Abate,"Automatic Speech Recognition for Amharic", P.HD. thesis, Dec.2005. Available: http://www.sub.unihamburg.de/opus/volltexte/2006/2981/pdf/thesis.pdf

[35] Yemane Keleta *et al*, "Nagaoka Tigrinya Corpus: Design and Development of Part-of-speech Tagged Corpus", unpublished

[36] Solomon Teferra Abate *et al*, "An Amharic Speech Corpus for Large Vocabulary Continuous Speech Recognition", unpublished

[37] B. H. Juang and L. R. Rabiner,"Hidden Markov Models for Speech Recognition", Technometrics, vol.33, pp.251-272, Aug.1991

[38] Li Deng and Dong Yu, "Interfacing DNNs with HMMs", in *Deep Learning Methods and Applications,* Foundation and Trends in Signal Processing, 2014.

[39] Mark Gales and Steve Young, "The Application of Hidden Markov Models in Speech Recognition", Foundations and Trends in Signal Processing, vol.1, pp. 195-304, 2007

[40] Daniel Povey *et al*. "The Kaldi speech recognition toolkit." *IEEE 2011 workshop on automatic speech recognition and understanding*. No. EPFL-CONF-192584. IEEE Signal Processing Society, 2011.

[41] Zhang, Xiaohui, et al. "Improving deep neural network acoustic models using generalized maxout networks." *Acoustics, Speech and Signal Processing (ICASSP), IEEE International Conference on*. IEEE, 2014.

[42] Joshua Meyer, "Kaldi Documentation", unpublished. [Online]. Available: http:// jrmeyer.github.io/

[43] Rath, Shakti P., et al. "Improved feature processing for deep neural networks." *Interspeech*. 2013.

[44] Goodfellow, Ian J. *et al*. "Maxout networks." *arXiv preprint arXiv: 1302.4389* , 2013.

[45] Gaida, Christian, et al. "Comparing open-source speech recognition toolkits." *Tech. Rep., DHBW Stuttgart* , 2014

[46] Doxygen. (2018, Feb.18). [Online]. Available:kaldi-asr.org/doc/dnn.html

[47] Piero Cosi *et al.* "A KALDI-DNN-based asr system for Italian." ,*International Joint Conference on* IEEE, 2015.

[48] Cosi, Piero, and Bryan L. Pellom. "Italian children's speech recognition for advanced interactive literacy tutors." Ninth European Conference on Speech Communication and Technology. 2005.

[49] Ondřej Plátek,"Automatic Speech Recognition using Kaldi",M.Sc. thesis,Charles University in Prague, 2014. Available: https://is.cuni.cz/webapps/zzp/download/120153012

Appendix A-Writing system of Tigrigna

	Ä	u	I	a	e	(ə)	o	wä	wi	wa	we	wə
H	ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ					
L	ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ					
ḥ	ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ					
M	መ	ሙ	ሚ	ማ	ሜ	ም	ሞ					
Ś	ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ					
R	ረ	ሩ	ሪ	ራ	ሬ	ር	ሮ					
S	ሰ	ሱ	ሲ	ሳ	ሴ	ስ	ሶ					
Š	ሸ	ሹ	ሺ	ሻ	ሼ	ሽ	ሾ					
ḳ	ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ	ቈ	ቊ	ቋ	ቌ	ቍ
ḳʰ	ቐ	ቑ	ቒ	ቓ	ቔ	ቕ	ቖ	ቘ	ቚ	ቛ	ቜ	ቝ
B	በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ					
V	ቨ	ቩ	ቪ	ቫ	ቬ	ቭ	ቮ					
T	ተ	ቱ	ቲ	ታ	ቴ	ት	ቶ					
Č	ቸ	ቹ	ቺ	ቻ	ቼ	ች	ቾ					
ḫ	ኀ	ኁ	ኂ	ኃ	ኄ	ኅ	ኆ	ኈ	ኊ	ኋ	ኌ	ኍ
N	ነ	ኑ	ኒ	ና	ኔ	ን	ኖ					
Ñ	ኘ	ኙ	ኚ	ኛ	ኜ	ኝ	ኞ					
ʾ	አ	ኡ	ኢ	ኣ	ኤ	እ	ኦ					
K	ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ	ኰ	ኲ	ኳ	ኴ	ኵ
X	ኸ	ኹ	ኺ	ኻ	ኼ	ኽ	ኾ	ዀ	ዂ	ዃ	ዄ	ዅ
W	ወ	ዉ	ዊ	ዋ	ዌ	ው	ዎ					
ʿ	ዐ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ					
Z	ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ					
Ž	ዠ	ዡ	ዢ	ዣ	ዤ	ዥ	ዦ					
Y	የ	ዩ	ዪ	ያ	ዬ	ይ	ዮ					
D	ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ					
Ǧ	ጀ	ጁ	ጂ	ጃ	ጄ	ጅ	ጆ					

G	ገ	ጉ	ጊ	ጋ	ጌ	ግ	ጎ	ጐ	ጒ	ጓ	ጔ	ጕ	
ṭ	ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ						
Č̣	ጩ	ጩ	ጪ	ጭ	ጭ	ጭ	ጭ						
P̣	ጰ	ጱ	ጲ	ጳ	ጴ	ጵ	ጶ						
ṣ	ጸ	ጹ	ጺ	ጻ	ጼ	ጽ	ጾ						
ṣ́	ፀ	ፁ	ፂ	ፃ	ፄ	ፅ	ፆ						
F	ፈ	ፉ	ፊ	ፋ	ፌ	ፍ	ፎ						
P	ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ						
	Ä	**u**	**I**	**a**	**e**	**(ə)**	**o**	**wä**	**wi**	**wa**	**we**	**wə**	

Appendix B- Font representation of Tigrigna

Tigrigna symbol	Ethiop font	Font used in this paper	Tigrigna example word
ብ	/b/	/b/	/bun/=’coffee’
ፕ	/p/	/p/	/pasta/=’pasta’
ጵ	/p’/	/P/	/p´et´ros/=’peter’
ድ	/d/	/d/	/dimmu/=’cat’
ት	/t/	/t/	/tolo/=’quick’
ጥ	/t’/	/T/	/t’af/=’tef’
ግ	/g/	/g/	/goggo/=’bread’
ክ	/k/	/k/	/kalbi/=’dog’
ቅ	/k’/	/q/	/k’al/=’word’
ኧ	/B/	/B/	/ʙadgi/=’donkey’
ፍ	/f/	/f/	/fik’ri/=’love’
ዝ	/z/	/z/	/zibʁi/=’hyena’
ስ	/s/	/s/	/siga/=’meat
ሽ	/š /	/x/	/šaš /=’veil’
ጽ	/s/	/S/	/s’om/=’fast’
ሀ	/h/	/h/	/hafti/=’wealth’
ሕ	/ħ/	/H/	/ ħaw/=’brother’
ዕ	/ɔ/	/J/	/ʕarki/=’friend’
ቾ	/č/	/c/	/ č iggir/=’problem’
ጭ	/č'/	/C/	/ č'aw/=’salt’
ጅ	/J/	/j/	/jigna/=’brave
ዥ	/Z/	/Z/	/does not exist/
ም	/m/	/m/	/ma?si/=’leather mat’
ን	/n/	/n/	/nega/=’ tomorrow’
ኝ	/ñ /	/N/	/ ñak/=’sound produced when a dog eating food’
ል	/l/	/l/	/lada/=’lamb’
ር	/r/	/r/	/ri?si/=’head’
ው	/w/	/w/	/waga/=’price’
ይ	/y/	/y/	/yeman/=’right-hand’

Appendix C- Pronunciation dictionary of the system

!SIL sil
<UNK> spn
BabBu B ab Bu
Babnnet B ab n et
Babrha B ab r h ha
Babti B ab t iy
Babzi B ab z iy
Bampul B am p ul
bluS b l uS
Bmbi B m b iy
brhan b r h an
Bti B t iy
Bwan B w an
Bwn B w n
Bxi B x iy
bzey b z ey
Camma C am ma
caynna C ay n na
CefCaf C ef C af
Cenna C en na
Cerqi C er q iy
Cew C ew
cok c ok
Crra C r ra
daNa d aN Na
degef d eg ef
derejja d er ej ja
deret d er et
flfl f l f l
flTet f l T et
frdi f r d iy
fSum f S um
genet g en et
genzeb g en z eb
gJzynna g J z y n na
glgallot g l g al lo t
gn g n
gujlle gu j l le
guJzzo gu J z zo
habeni h ab en iy
Hadde H ad de
Hadux H ad ux
hafti h af t iy
hager h ag er
Hamsa H am s sa
Hamuxtte H am ux t te
HaqeNNa H aq eN Na
haymanot h ay m an ot
Hbret H b r et
hdmmo h d m mo
Hgi H g iy
Hgus H gu s

Hlbbet H l b et
Hlmi H l m iy
Hmbaxxa H m b ax xa
hnSSa h n S Sa
Hray H r ay
hSan h S an
hzbi h z b iy
ityoPya iy t y op ya
Jadi J ad iy
Jammet J am me t
jebenna j eb en na
jgnna j g n na
Jnqfat J n q f at
Jrre J r re
kab k ab
kabti k ab t iy
kara k ar ah
kfal k f al
kfli k f l iy
kll k l l
klte k l t te
korecca ku or ec ca
kremti k r em t iy
ksab k s ab
kulu ku l uw
laHmi l aH m iy
laPis l aP iy s
lJli l J l iy
lmdi l m d iy
lmJat l m J at
maBger m aB g er
maJrre m aJ r re
may m ay
mBnti m B n t iy
mcw m c w
meCerexxa m eC er ex xa
medeb m ed eb
meJar m eJ ar
mejemerya m ej em er ya
melHas m el H as
memhr m em h r
memrHi m em r H iy
mengsti m en g s t iy
menqeNNa m en q eN Na
mesarHi m es ar H iy
mesel m es el
meSHaf m eS H af
mesmer m es m er
meTen m eT en
metkel m et k el
mhzzo m h z zo
mJraf m J r af
mqlas m q l as
mrggaS m r g g aS

nab n ab
nabrra n ab r ra
nebar n eb ar
negereNNa n eg er eN Na
ngdi n g d iy
PaPas P aP as
papayye p ap ay ye
pastta p as t ta
Petros P et r os
polis p ol iy s
postta p os t ta
prezidant p re z iy d an t
qbul q bu l
qdmi q d m iy
qlTuf q l T uf
qSJat q S Ja t
quTebba qu T eb ba
rBsi r B s iy
seb s eb
Sebba S eb ba
sebay s eb ay
SebSab S eb S ab
selestte s el es t te
slezi s l ez iy
sraH s r aH
Tebay T eb ay
Tebenjja T eb en j ja
tefeTrro t ef eT r ro
tegeliSu t eg el iy Su
tegbar t eg b ar
temharay t em h ar ay
temokrro t em ok r ro
Tenkarra T en k ar ra
TerePezza T er eP ez za
tgray t g r ay
tgrNNa t g r N Na
tmhrti t m h r t iy
Tjnna T J n na
tkal t k al
tlmi t l m iy
trfi t r f iy
trgum t r gu m
txJatte t x Ja t te
weredda w er ed da
wereNNa w er eN Na
weyanne w ey an ne
wHj w H j
wlad w l ad
xduxtte x d ux t te
xemontte x em on t te
xewJatte x ew Ja t te
xfan x f an
xgr x g r
xH x H

xlmat x l m at
zelewwu z el ew wu
zerebba z er eb ba
zlla z l la

Appendix D: The adapted Kaldi scripts for GMM-HMM, DNN-HMM training and Decoding

Monophone training and alignment

Monophone training:

steps/train_mono.sh --nj 4 data/train data/lang exp/mono

where nj is the number of jobs to run in parallel for training the model.

Monophone alignment:

steps/align_si.sh --nj 4 data/train data/lang exp/mono exp/mono_ali

Triphone Training and alignment

Training delta based triphones:

steps/train_deltas.sh 2000 11000 data/train data/lang exp/mono_ali exp/tri1

Aligning delta based triphones:

steps/align_si.sh --nj 4 data/train data/lang exp/tri1 exp/tri1_ali

Training delta+delta-delta triphones:

steps/train_deltas.sh 2000 11000 data/train data/lang exp/tri1_ali exp/tri2a

Aligning delta+delta-delta triphones:

steps/align_si.sh --nj 4 data/train data/lang exp/tri2a exp/tri2a_ali

Training LDA-MLLT triphones:

steps/train_lda_mllt.sh 2500 15000 data/train data/lang exp/tri2a_ali exp/tri3a

Aligning LDA-MLLT triphones with FMLLR:

steps/align_fmllr.sh --nj 4 data/train data/lang exp/tri3a exp/tri3a_ali

Train SAT triphones:

steps/train_sat.sh 2500 15000 data/train data/lang exp/tri3a_ali exp/tri4a

Align SAT triphones with FMLLR:

steps/align_fmllr.sh data/train data/lang exp/tri4a exp/tri4a_ali

DNN-HMM training

steps/nnet2/train_tanh.sh data/train data/lang exp/tri3_ali exp/tri4_nnet

where, *tri3-ali* is a phone-to-audio alignment taken from the result of GMM-HMM training