ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

COLLEGE OF NATURAL SCIENCES

FACULTY OF COMPUTER AND MATHEMATICAL SCIENCES

DEPARTMENT OF COMPUTER SCIENCE

AMHARIC TEXT-PROMPTED SPEAKER VERIFICATION MODEL

BY

WOLELAW BERHANE ALEMU

A THESIS SUMITTED TO

THE SCHOOL OF GRADUATE STUDIES OF ADDIS ABABA UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF MASTERS OF SCIENCE

IN COMPUTER SCIENCE

MAY 2011

ADDIS ABABA UNIVERSITY

SCHOOL OF GRADUATE STUDIES

COLLEGE OF NATURAL SCIENCES

FACULTY OF COMPUTER AND MATHEMATICAL SCIENCES

DEPARTMENT OF COMPUTER SCIENCE

AMHARIC TEXT-PROMPTED SPEAKER VERIFICATION MODEL

BY

WOLELAW BERHANE ALEMU

ADVISOR:

SEBSIBE HAILEMARIAM (PhD)

APPROVED BY

EXAMINING BOARD

1. Dr. Sebsibe H/Mariam, Advisor _____
2. _____ _____
3. _____ _____

Dedication

To my mother whom I lost at the age of mine!
And to my father who has become a monk thereafter
but committed to upbringing me.

Acknowledgements

First and foremost I would like to thank my advisor Dr. Sebsibe Hailemariam for his dedicated professional guidance throughout this thesis work. And also, this is the only opportunity I got to thank the Department of Computer Science for the chance I was given to join the M.SC program. I appreciate class mates for the relationships maintained since then. I would also like to thank work mates and external colleagues for the hard times.

Finally, here comes the family who were always motivating me to work on the thesis, thesis and thesis!

Table of Contents

List of Figures

List of Tables

List of Acronyms

Acronym	Description
ANN	Artificial Neural Network
ATM	Automatic Teller Machine
ATPSVM	Amharic Text-Prompted Speaker Verification Model
DCT	Discrete Cosine Transform
DNA	Deoxyribonucleic acid
DTFT	Discrete Time Fourier Transform
DTW	Dynamic Time Warping
EER	Equal Error Rate
FA	False Acceptance
FAR	False Acceptance Rate
FFT	Fast Fourier Transform
FR	False Rejection
FRR	False Rejection Rate
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
ID	Identification code
IDFT	Inverse Discrete Fourier Transform
IDTFT	Inverse Discrete Time Fourier Transform
IEEE	Institute of Electrical and Electronics Engineers
IMECS	International MultiConference of Engineers and Computer Scientists
LP	Linear Prediction
LPC	Linear Prediction Coding
LPCCs	Linear Prediction Cepstral Coefficients
MFC	Mel-Frequency Cepstrum
MFCCs	Mel Frequency Cepstral Coefficients
nMFCCs	Normalized MFCCs
MLPs	Multi-Layered Perceptrons
NIST	National Institute of Standards and Technology
OS	Operating System
PIN	personal identification numbers
PLP	Perceptual Linear Prediction
PLPCCs	Perceptual Linear Prediction Cepstral Coefficients

RBF	Gaussian Radial Basis Function kernel
RIFF	Resource Interchange File Format
SMS	Short Message Service
SVM	support vector machine
TA	True Acceptance
TR	True Rejection
UBM	Universal Background Model
VQ	Vector Quantization

List of Annexes

Abstract

Speaker verification is a biometric authentication model that takes speech signal as an input to verify a claimed speaker. A speaker verification model extracts speaker dependent characteristics from the speech wave signal so as to create the voiceprint of the speaker. The researcher has implemented the logic in an English speaker verification model [45] for Amharic. But the average accuracy obtained for ten Amharic words is 92.93%. The research work is initiated from this implementation and its subsequent poor accuracy performance. In this thesis research work, Amharic Text-Prompted Speaker Verification Model (ATPSVM) is designed and implemented.

The ATPSVM model applies frame-based processing to the speech wave signals so that all samples in a frame are processed simultaneously. It extracts speaker feature vectors as Mel Frequency Cepstral Coefficients for use in speaker model construction. Then it applies the parameter domain (spectral) normalization followed by the min-max normalization on the speaker feature vectors so as to scale the feature vector values in [0, 1]. Finally, it applies Support Vector Machine kernel functions for modelling each speaker. For a specific Amharic word prompted, it utilizes one-against-each SVM speaker modeling strategy to maintain the balance of the test speaker feature vectors in the mixed features.

The ATPSVM model prototype is evaluated using ten Amharic words. Each Amharic word is uttered ten times repeatedly by 5 men and 5 women. So that a total of 100 Speech wave files are recorded from each speaker.

One utterance is iteratively taken for testing while the remaining 9 are used for training the speaker on leave-one-out basis. It iteratively takes one utterance of each speaker against other 9 speakers for testing. The respective

utterance of other speaker is taken as impostor data set for the same. The remaining respective 9 utterances from the two speakers are taken as training data sets.

Thus for each Amharic word, the model is evaluated using 900 training data sets, 900 client testing data sets and another 900 impostor testing data sets. In total, the ATPSVM model is evaluated using 9,000 training data sets, 9,000 client (target) testing data sets, and another 9,000 impostor testing data sets. The evaluation of the model is done by varying the threshold *theta* between 0.0 and 1.0 with 0.1 differences.

Performance is measured in terms of False Acceptance, False Rejection, True Acceptance and True Rejection. Then it is reported as Precision, Accuracy, Recall, False Acceptance Rate, False Rejection Rate and Equal Error Rate (EER). The ATPSVM model is evaluated using the SVM Linear, Gaussian Radial Basis Network (RBF), Multilayer Perceptron (MLP), and Polynomial power 3 kernel functions. Best performance of the ATPSVM model is obtained when the SVM Polynomial Power 3 kernel function is applied. The performance difference between the kernel functions follows from the algorithmic definition of the same.

Using the SVM Polynomial Power 3 kernel function, for the ten Amharic words experimented, an average performance of 0.25% EER, 99.7% accuracy, 99.8% recall and 99.7% precision is obtained. For the same kernel, the performance of the ATPSVM model for each Amharic word is also evaluated separately. For 70% of the Amharic words experimented, the performance of the model is 0.00% EER with 100% accuracy, 100% precision, and 100% recall values. For the remaining 30%, its performance is slightly lower than these values. By selecting more discriminative Amharic words of similar nature to the seven words, it is possible to get the desired highest performance from the ATPSVM model.

Keywords: Speaker Verification, Speaker Recognition, Biometric Authentication, Amharic Text-Prompted Speaker Verification, Text-Prompted Speaker Verification.

1. INTRODUCTION

1.1 Overview

Biometrics is an emerging technology for automatically identifying individuals using their distinct physical or behavioural characteristics such as fingerprint, voice, face, iris, retina, palm, signature, wrist vein, keystroke dynamics, DNA (Deoxyribonucleic acid) and hand geometry [1]. A comparison of these biometric authentications is presented in section 1.2. Biometric authentication provides a better solution for the increased security requirements of our information society than current identification methods such as passwords, personal identification numbers (PIN) and tokens for various reasons. One reason is the person to be identified must be physically present at the point of identification. In addition, it avoids the need for the person to remember a password or PIN or carry a token. For a heightened security, biometric authentications can also be used along with other security methods such as PIN, tokens, username and password [1].

Speaker recognition is the process of automatically recognizing who is speaking by using the speaker-specific information included in speech waves to verify identities being claimed by people accessing the systems; that is, it enables access control of various services by voice [2]. It is different from speech recognition because the speaker recognition technology does not recognize the spoken word itself. Rather, it recognizes the speaker of the words by analyzing unique speaker dependent speech characteristics which are mainly dependent on individual spectral envelope (vocal tract shape) and voice source differences [2]. A speaker recognition system requires to train or enroll the speaker in the system so that the system can characterize (or learn) the voice patterns of that specific speaker. The system generates a secure, encoded voiceprint (model) of the speaker. So, each speaker will have a corresponding characterizing voiceprint in the system. Then during recognition when the speaker calls back, the speaker specific characteristics of the new utterance are measured for

feature vectors similarity with the pre-recorded voiceprint. We can visualize a speaker recognition system as a mapping table that maps a speaker into a unique voiceprint (speaker model).

Speaker recognition task can be divided into speaker verification and speaker identification [2]. Speaker Verification is the science of verifying a person's identity on the basis of his/her voice characteristics and it serves as a confirmation to the identity of a particular person so as to access secured equipment and restricted services such as phone access to banking and shopping. When a Speaker ID J and speech Voice K are given together as inputs to the system, the speaker verification system has to verify whether the speaker (Speaker ID J) is really characterized by the voice (Voice K). The expected output is either rejection or success which is a binary decision. In Speaker identification, when Voice K is given as an input to the system, it identifies among the N trained (enrolled) voiceprints, may be after N similarity measures, to which speaker Voice K is best mapped. It predicts the owner of the voice or the case may be that voice K does not match any of the models (open-set identification). In verification, the performance does not depend on the number of enrolments where as in identification it decreases with increase in enrolment population [2]. The reason is the number of similarity measures; which is only one in speaker verification and the number of population in speaker identification. In both speaker verification and identification, additional threshold test is applied to the speech signal to determine whether the match is close enough to accept the decision or to ask for a new trial.

1.2 Comparison of biometric authentications

1.2.1 Speaker verification biometrics

In speaker verification biometrics, the unique features of a person’s voiced speech sounds are digitized and compared with the individual’s pre-recorded speaker model (voiceprint) stored in the database for identity verification.

1.2.2 Fingerprint biometrics

In fingerprint authentication, individual uniqueness is determined by the pattern of ridges and furrows as well as the minutiae points that are found on the surface of the finger [1, 3]. In the past, fingerprints were recorded by the application of ink to the finger which was then pressed to paper to give an impression. More recently, it has become possible to scan a person's fingerprint into virtual storage in a computer with the aid of laser technology. During verification, a person's fingerprint will be scanned again by a similar device, and a match of print to name is verified through information systems. Fingerprint verification is currently being used in applications such as automatic teller machines (ATM), Ethiopian Immigration System, Ethiopian Taxing System, and as security locks in latest laptops. But digital fingerprinting is ineffectual with the handicapped and those who have no limps.

1.2.3 Iris and retina biometrics

In iris and retina biometrics, the complex and unique characteristics of the eye are analyzed and used for identity verification [1, 3]. The iris is the coloured band of tissue that surrounds the pupil of the eye. An iris biometrics system uses a video camera to capture the sample while the software compares the resulting data against stored templates. The retina is the layer of blood vessels at the back of the eye. Retina biometrics is performed by directing a low-intensity infrared light to capture the unique retina characteristics. The centre of the retina is scanned and the unique pattern of the blood vessels is captured which is inconvenient, intrusive and difficult to gain general acceptance by the end user. Iris and retinal biometric systems are ineffectual with the blind and those who have cataracts.

1.2.4 Face recognition biometrics

In face recognition systems, an individual is recognized by analyzing the unique shape, pattern and positioning of facial features [1, 3]. This biometric system reduces the amount of man-machine interaction time. However, this system is highly

unreliable and expensive. For example, it will not distinguish twins or triplets and it does not verify the user after a haircut, and it does not verify a person who changes from wearing and not wearing glasses [1].

1.2.5 Signature verification biometrics

In a signature verification system, the user signs his signature on a digitized graphics tablet. Signature dynamics, such as speed, relative speed, stroke order, stroke count and pressure are analyzed. The key in signature dynamics is to differentiate between the parts of the signature that are habitual and those that vary with almost every signing and hence a dynamic signature verification system has to be engineered in a way to adapt to such variances [1, 3]. Thus, the problems with signature verification lie in the means of obtaining the measurements used in the verification process and the repeatability of the signature and of course not applicable with the handicapped. Currently, Ethiopian Immigration System utilizes signature verification for passport renewal.

1.2.6 Keystroke dynamics biometrics

In keystroke dynamics biometric authentication, an individual is authenticated by analyzing the way he or she types on keyboard [1, 3]. It is a very new technology to the biometrics arena. Users enrol by typing the same word a number of times. Verification is based on the concept that the rhythm with which one types is distinctive [1].

1.2.7 DNA biometrics

In DNA (Deoxyribonucleic acid) biometric authentication, an individual's DNA is tested. This authentication is a very accurate way of proving identity as DNA is completely unique for every person [1, 3]. DNA is found in every cell of every creature, and it contains the information for carrying out the activities of the cell. Due to the extensive testing and advanced technology required, it is very expensive, but when a positive verification is needed it is the most reliable [1].

1.2.8 Biometrics Comparison

Comparison of biometric authentications voice, iris, face and fingerprint for identity verification is made in [3] in terms of:

- *Distinctiveness* (existence of wide differences in the pattern among the population),
- *Robustness* (repeatable and not subject to large changes over time),
- *Accessibility* (easily presentable to a sensor), and
- *Acceptability* (perceived as non-intrusive by the users).

Table 1.1: Comparison of Biometric authentications for identity verification

Property	**Face**	**Speech voice**	**Iris**	**Fingerprint**
Distinctiveness	High	Moderate	High	High
Robustness	High	Moderate	High	Moderate
Accessibility	Moderate	High	Low	Moderate
Acceptability	High	High	Moderate	Moderate

According to [3], all information security measures including biometric authentications try to address at least one of the following three goals;

- Protecting the confidentiality of data,
- Protecting the integrity of data, and
- Promoting the availability of data for the authorized users.

Thus, high accessibility and high acceptability of human voice from Table 1.1 promotes the availability of secured data for the authorized users authenticated through their speech voice at any time remotely from anywhere.

1.3 Speaker verification as a security system

Fast, efficient, accurate, and robust means of recognizing people is of growing importance for commercial, forensic, and government applications. Human voice can serve as a key for any security objects, and it is not so easy in general to lose or forget

it and can be transmitted by telephone channel [4]. This provides a speaker verification system an ability to automatically verify speakers and provide access to security objects through mobile phone remotely.

Speaker verification is an enhanced voice self-service application. It is an extension of an organization's security policies. It has become the necessary authentication process to front every security-conscious conversation [5].

The goal of a speaker verification system is to have a fool proof system with the lowest imposter success rate and to enhance the customer experience and increase overall customer satisfaction without sacrificing convenience. Several researches have been conducted on speaker verification model. Table 1.2 summarizes the different experimental scenarios and results obtained.

Table 1.2: Summary of speaker verification experimental scenarios and results

Language	Feature	Model	Dataset	Performance	Method
French [12]	LPCC	SVM	Yoho (English) for training, PolyVar (French) for testing	EER of 4.3%	Text-independent
Romanian [36]	MFCC	HMM	Romanian corpus from 11 People	1% EER	Text-prompted
Malay [37]	MFCC	ANN/MLP	Corpus of 30 speakers; 20 for training and 10 for testing	74% for 30, 80% for 20, and 90% for 10 speakers.	Text-independent
Arabic [38]	MFCC	GMM	Saudi Accented Arabic Voice Bank, 1033 speaker	4.58% EER	Text-independent
Romanian [39]	PLPCC	HMM/SVM	10 speakers corpus	Minimum EER of 12%.	Text-independent
Spanish [40]	MFCC	HMM	98 clients from 184 speakers corpus	2.10% FA, 2.26% FR.	Text-prompted

Swiss German [41]	LPCC	HMM/MLP	HER corpus of 536 speakers	EER of 8.1%	Text-prompted
English [42]	PLPCC	SVM	2002 NIST SRE	7.31% EER	Text-independent
Malay [43]	LPCC	VQ/HMM	100 speakers Malay spoken database	11.72% EER.	Text-dependent
English [44]	LPCC	SVM	30 speakers corpus	2.12% EER	Text-dependent
English [45]	MFCC	SVM	20 women and 20 men speakers' corpus.	0.00% EER & 95.1% average accuracy rate.	Text-dependent

From Table 1.2 the work in [45] reports the best performance of EER at some specific English text(s) and average accuracy of 95.1%. The work uses Support vector machine (SVM) model that generates one SVM structure per speaker. This SVM structure is trained in such a way that each training data comprises of one speaker true samples and all other impostors as false samples. That is one-against-all scenario.

1.4 Characteristics of a speaker verification system

A speaker verification system has the following major characteristics:

- It acts as a checking gate for the speaker to reach at the service provider and use the desired service.
- It can take verification voice content remotely and provide verification of a person with long distance capabilities.
- It is cost-effective, user friendly and convenient to use. One can use only his or her mobile phone and speech. Speech is a natural signal to produce and not considered non-invasive by users. The mobile phone provides the speech signal of the user to the verification system.

- It requires each user to create its own speaker model through voice enrolment (training) before using the voiceprint verification task. That is, a speaker verification system works in two phases. During the first (training) phase, users enroll in the system by speaking a set of words one or more times. For example, they might read alphanumeric phrases. From this speech wave, the system generates a secure, encoded voiceprint (speaker model). During the second (verification) phase, an enrolled user calls back into the system. Then the user is supposed to speak something or asked to repeat a word or a phrase. From what is spoken, user identity dependent characteristics are extracted, modeled and similarity measure is done to the enrolled voiceprint. If the match is as good as to a certain threshold, the user is granted access for the service; if it is poor, the user is denied for the service.
- It can be easily adjusted to account for multimodal authentication with minimal user burden and no user hardware requirements.

1.5 Application areas of speaker verification systems

An automatic speaker verification system can be applied in various kinds of security systems together with other authentication mechanisms. Some are as summarized below:

- *Mobile Banking*: Mobile banking serves the customer to do an automatic banking transaction or complementary to the standing instructions of banking operations remotely over a mobile telephone network. It acts as a delivery channel for the core banking system. The customer is provided secured access to his/her bank account through mobile phone authenticated by his/her speech voice.
- *Legal and Court Cases system:* A speaker verification technology can be applied to Legal and Court Cases system so that a user of the system can access his/her court case remotely through telephone authenticated by voice so as to know next appointment date and other court decisions & progresses.

- *Password-free OS/Application login*: Instead of remembering long and complex passwords or carrying tokens to log into an operating system or an application, speaker verification system can be used as authentication for login.
- *On-site Entrance control:* A speaker verification system can be used to prove identity during access to any physical facilities like home incarceration by storing speaker model in a small chip.
- *Shopping*: A speaker verification system can be used to authenticate an online customer shopping through mobile phone.

1.6 Motivation to work on speaker verification

In today's world there is a growing concern regarding identity theft, national security, and on-line terrorism [4] where as the rapid advances in networking, communication, mobility and the associated strong concerns of security entail the need for reliable user authentication techniques. The rapid growth of mobile phone users all over the world confirms that the human voice is the most accessible biometric, as no extra acquisition device or transmission system is needed. This fact gives human speech voice an advantage over other biometrics especially when remote users of computer applications are taken into account [5].

Banking customers around the world are increasingly demanding easier and more instantaneous access to their accounts. Direct channels such as mobile banking are the norm and provide customers with 24/7 banking services. But identity is usually verified with a PIN (personal identification number) along with a series of security questions and account details. However, these details can be forgotten and this can cause customer inconvenience and increased costs. In addition, some customers may expose themselves to the risk of fraud or identity theft if they store identity information in a diary, wallet or handbag. Biometric speech voice verification addresses these problems [1, 6].

Currently some Ethiopian banks are on the process of implementing mobile banking as a delivery channel for the banking services. So, one practical motive is that

such a customer needs to have a speaker verification system designed in Ethiopian context where local languages can be used for authentication. In this case, speaker verification is preferred to speaker identification because the performance in identification decreases with increase in the number of enrolments. Then the customer can benefit from speaker verification technology for more accurate, secured and remote account access through the mobile phone call. In addition to mobile banking stakeholders, many other service giving organizations such as courts, ticket offices, etc are also beneficiaries of a speaker verification system with the system adjusted as per the respective environmental security level. Thus, it is worth focusing this thesis work on speaker verification.

Table 1.2 shows speaker verification models developed for different native languages and [45] has reported the best performance among them with 0.00% EER and 95.1% average accuracy. Lowest EER means highest performance. But this English model, [45], cannot be applied in Ethiopia as some Ethiopians do not speak English. So a model that works in native Ethiopian Languages should be implemented.

The researcher has initially attempted to implement the speaker verification model reported in [45] for Amharic with the same logic. So that one SVM structure is generated for each speaker against all others for ten Amharic words. The performance result is as in Table 1.3.

Table 1.3 Result after [45] is implemented and experimented for Amharic

Amharic word Code	Accuracy (%)
1	90.21
2	90.05
3	98.06
4	90.05
5	99.10
6	90.65
7	92.25
8	91.93
9	90.89

10	96.09
Average	**92.93**

Table 1.3 shows that the average accuracy of the model in [45] when implemented for Amharic has become 92.93%, even lower than the English one.

In [45], to train model of a speaker, it has applied one-against-all strategy of SVM classification. In one-against-all strategy, as the number of enrolled speakers increases the balance of test speaker feature vectors to that of all others in the mixed features decreases. This could be the reason for the accuracy gap of the model, [45], why its accuracy is much less than 100%.

One speaker against each other speaker (one-against-each) SVM classification method can maintain the balance for the test speaker feature vectors in the mixed features. Therefore, it is important to do a research work so as to improve the accuracy of the speaker verification system in Amharic.

1.7 Statement of the Problem

From Table 1.2, [45] has reported best performance of 95.1% average accuracy. The researcher's implementation of the model in [45] for Amharic has reported an average accuracy of 92.93%. For secured applications like mobile banking this accuracy is not good enough. The accuracy has to be closer to 100%. So it is vital to find out the root cause of the accuracy performance gap for the model. Finding the root cause enables to improve the accuracy of speaker verification in Amharic. The detail of the model in [45] will be discussed in the subsequent Chapters.

1.8 Research Questions

How can the model in [45] be modified so as to improve the accuracy of speaker verification in Amharic?

1.9 Research Objectives

1.9.1 General Objective

The general objective of this research work is to design and implement a speaker verification model in Amharic with accuracy closer to 100%.

1.9.2 Specific Objectives

The specific objectives of this research work include:

[1]. To propose modified speaker verification model.

[2]. To select appropriate features for the proposed new model.

[3]. To apply appropriate feature normalization technique.

[4]. To select appropriate tools for the design and implementation of the model.

[5]. To develop the Amharic Text-Prompted Speaker Verification model Prototype.

[6]. To evaluate the performance of the model.

1.10 Scope and limitation of the research

Speech will be recorded in a normal environment on a laboratory machine. But this may not necessarily be the case in most real world applications. Phone based speeches may behave differently. The channel and the telephone equipment may affect speech quality. However, such data performance may not vary significantly. The reason for not taking telephone speech is because of lack of resources.

1.11 Research Methodology

The methodology to be applied in this study involves the following tasks:

1.11.1 Research background

- Detail literature review will be done on human speech production system, speech analysis, speaker feature extraction techniques, speaker modelling algorithms, and decision making in relation to speaker verification.
- Related works will be reviewed and the respective methodologies applied in each work will be analyzed.

1.11.2 Data collection

- A list of Amharic words will be selected for training and testing the model.
- A set of speakers who will utter these words will be selected.
- The speech wave files of the Amharic words will be recorded from the speakers.
- Each speaker will utter each Amharic word repeatedly and the respective utterance wave file will be recorded separately.

1.11.3 Speaker feature selection

This thesis work will apply MFCCs to extract feature vectors from Amharic speeches of a speaker.

1.11.4 Speaker modelling techniques

This research will apply SVM as speaker modelling technique.

1.11.5 Tools and Techniques

- Wavesurfer 1.8.8 will be used for recording speech wave files.
- Matlab 7.6 will be used for developing the prototype of the model.
- Performance will be reported using tables and sketched graphs.
- Ms Excel will be applied for storing and manipulating intermediary speech voice feature vector data to be used for training and testing.
- Experimental result analysis will be done using MedCalc11.5.1.0 and Ms Excel.

1.11.6 Testing Methodology

- Leave-one-out testing approach will be adopted for the research as it provides reliable test results for proper result analysis.
- Performance of the model will be measured using different metrics of biometrics and different cut-off points for decision. This includes false

acceptance, false rejection, true acceptance true rejection and the aggregate metrics like EER, FAR, FRR, Accuracy, Recall, and Precision.

1.12 Thesis organization

The thesis is organized in seven chapters including the current chapter. Chapter 2, Literature Review on Speaker verification, lays the foundation what speaker verification is all about. It is in this Chapter where the speech production and speech analysis are discussed. The main speaker feature extraction and speaker modelling techniques are also well described. Chapter 3 discusses the feature and model selection for the speaker verification model in Amharic. Chapter 4 focuses on the design aspect of the thesis work. It presents the architecture of the ATPSVM model and its algorithms on feature extraction, speaker modelling and verification decision making. Chapter 5 presents the implementation of the ATPSVM model. Chapter 6 describes ATPSVM model performance evaluation metrics, detail evaluation methods and steps. Finally it reports results in tables and sketched graphs. Chapter 7 concludes the research work and gives an insight to future works.

2 REVIEW OF LITERATURE

2.1 Introduction

In this Chapter, core concepts related to the thesis work are covered. It describes speech analysis, fundamental components of speaker verification, speaker verification methods, speaker feature extraction and speaker modelling techniques. It is the launching pad to devise the methods and techniques for the design of the desired ATPSVM model.

2.2 Speech production and Analysis

Speech is used to communicate information from a speaker to a listener. It is the most natural means of communication for almost everyone. Speech analysis is concerned with processing techniques that are designed to extract information from the speech waveform. It is the core task for a speaker verification process. Understanding the characteristics of the human sound production (phonetics) is an important input for speech analysis.

The human speech voice, the key to speaker verification, is produced as a combined effect of the lungs, the vocal cords and the articulators [7]. The vocal folds (vocal cords) within the larynx are the primary sound source. They are a vibrating valve that chops up the airflow from the lungs into audible pulses that form the laryngeal sound source. A person's pitch frequency also originates in the vocal cords/folds; it is the rate at which the vocal folds vibrate [7]. The lung produces adequate airflow and air pressure to vibrate the vocal folds. The muscles of the larynx adjust the length and tension of the vocal folds to fine tune pitch and tone. The articulators (the parts of the vocal tract above the larynx consisting of tongue, palate, cheek, lips, teeth, etc.) articulate and filter the sound emanating from the larynx. They interact with the laryngeal airflow and hence their physical condition (shape and size) controls the voice quality [7]. The vocal folds in combination with the articulators are capable of producing highly intricate arrays of sound for talking, singing, laughing and crying

[7]. Figure 2.1 illustrates the vocal tract (shaded area), the articulators and other body organs involved in speech production.

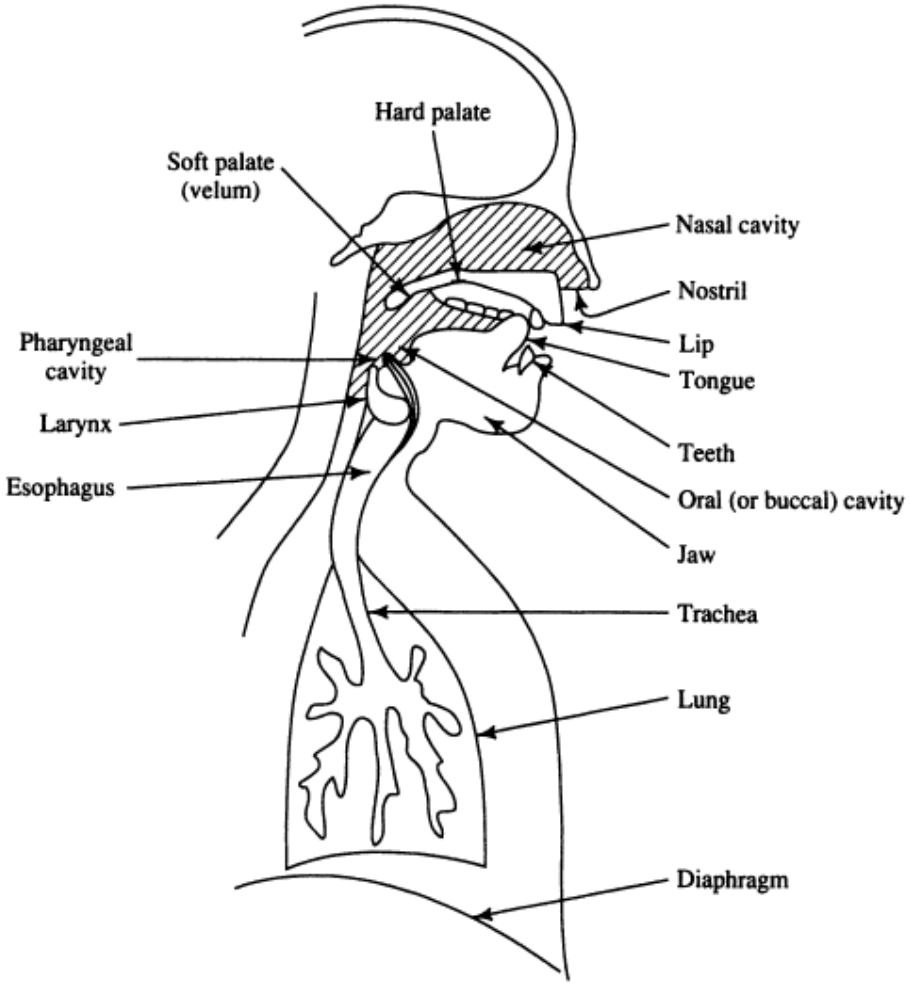

Figure 2.1: The vocal tract (shaded) and speech production system

As a physiological aspect of the voice biometrics, no two individuals sound is identical as their vocal tract shapes, larynx sizes, and other parts of their voice production organs are different [7, 8]. In addition to these physical (vocal tract) differences, as a behavioural aspect of the voice biometrics each speaker has his or her characteristic manner of speaking, including the use of a particular accent, rhythm, intonation style, pronunciation pattern and choice of vocabulary [5, 7].

As depicted by the shaded area in Figure 2.1, the vocal tract is generally considered as the speech production organ above the vocal folds consisting of [5, 7]:

- *laryngeal pharynx* (beneath the epiglottis),

- *oral pharynx* (behind the tongue between the epiglottis and the velum),
- *oral cavity* (forward of the velum bounded by the palate, tongue and palate),
- *nasal pharynx* (above the velum just the rear end of the nasal cavity), and;
- *nasal cavity* (above the palate and extending from the pharynx to the nostrils).

It is about 17cm in length for an adult man while it is 14cm for a woman [7]. Repositioning of the articulators causes the cross-sectional area of the vocal tract to vary along its length from zero (complete closure) to greater than $20cm^2$ [7]. The nasal tract constitutes an auxiliary path for the transmission of sound. A typical length for the nasal tract in an adult male is 12cm [7]. Acoustic coupling between the nasal and vocal tracts is controlled by the size of the opening at the velum.

During speech production, the vocal tract modifies the spectral content of the acoustic wave as it passes through [5, 7, 8]. And it is common in speaker verification systems to use features derived from the vocal tract as it has static and dynamic distinctive features [5, 7].

2.2.1 Speech sound types

Speech sounds can be classified as voiced and unvoiced [7]. Voiced speech signal is characterized by deterministic acoustic waveforms, while unvoiced speech signal corresponds to stochastic waveforms [7]. Voiced sounds are produced by forcing air through the glottis or an opening between the vocal folds [7]. The tension of the vocal cords is adjusted so that they vibrate in oscillatory fashion. The periodic interruption of the sub-glottal airflow results in quasi-periodic puffs of air that excite the vocal tract [7]. Voice or phonation is sound produced by the larynx. The larynx provides a periodic excitation to voiced speech sounds [7].

Unvoiced sounds are generated by forming a constriction at some point along the vocal tract, and forcing air through the constriction to produce noise like turbulence [7]. A speech sound may be voiced and at the same time constricted (mixed) [7]. Voiced speech sounds have received more research attention than the unvoiced case

[7]. This is due to studies in speech perception that suggest accurate modelling of voiced speech is crucial for natural-sounding speech [7].

Speech sound wave is modelled as the output of linear, time-varying system excited by either quasi-periodic pulses (during voiced speech), or random noise (during unvoiced speech) [7].

A voiced speech signal y(n) is composed of a quickly varying part x(n) (excitation sequence) convolved with a slowly varying part h(n) (vocal system impulse response) as [7, 9]:

$$y(n)=x(n)* h(n) \quad (1)$$

where * characterizes the linear convolution operator. This is time-domain representation for a frame of voiced speech ending at time n. The claimed speaker is verified using the pattern of the vocal system impulse response characteristics h(n) [7, 9]. So h(n) is the focus of the speaker verification process. But the convolution makes it difficult to separate the two parts x(n) and h(n) [7, 9]. Hence separating them is an important step of the speaker verification process.

2.3 Speaker Verification Methods

A speaker verification system is a security system to allow users to gain access to a service. Its goal is to certify (attest) the claimed identity of a user. It is increasingly often used to secure personal information, particularly for mobile phone based applications [9]. It enables to distinguish, in real time, between various speakers based only on the input speech signal from the claimed speaker.

A speaker verification system performs a binary classification task. The computer system allows the user to get access to the protected service if and only if it recognizes the user from their speech waveform. This enables organizations to deploy automated intelligence services using voice. For instance, Ethiopian banks can secure their mobile banking service using a speaker verification system. It is implemented using different methods.

According to [10], speaker verification methods can be categorized as text-dependent and text-independent approaches. In the text-dependent case, the utterance presented by the speaker is known by the speaker verification system beforehand. In the text-independent case, no assumption about the text being spoken is made, but the system must model the general underlying properties of the speaker's vocal space. In general, text-dependent systems are more accurate, since the speech voice is going to be compared with to an existing model of the same [10]. Text-dependent approach in turn can be categorized as fixed-text and text-prompted speaker verification methods [10]. So a given speaker verification system can deploy any one amongst fixed-text, text-prompted and text-independent methods.

2.3.1 Fixed-text method

In this approach the text in both training and testing is the same and is known. Fixed-text methods tend to give the best speaker verification performance because the text is known and hence these methods can fully exploit the speaker individuality that is associated with each sound in the utterance [10]. But this fixed-text is vulnerable to attack. So it is important to guard against attack by someone playing back a suitable voice recording of an authorized speaker. It is used for applications with strong control over user input and where the linguistic content of the speech is known [10].

2.3.2 Text-prompted method

In text-prompted approach the user is prompted to enter a sequence of key words that is chosen randomly every time the system is used. In text-prompted speaker verification, the system must both recognize the identity of the speaker and verify that the utterance is indeed a spoken version of the specified text [10]. Thus an utterance should be rejected if its text differs from the prompted text, even if it is spoken by the registered speaker. The advantage of text prompting is that it avoids the fixed-text problem in the fixed-text approach. A possible intruder cannot know beforehand what the phrase will be, and playback of pre-recorded speech becomes difficult.

Furthermore, the system can be made the user to utter the pass phrase within a short time interval, which makes the intruder harder to use a device or software that synthesizes the customer's voice. Text-prompted speaker verification methods are used for applications with strong control over user input and where users are cooperative and the linguistic content of the speech is known [10].

2.3.3 Text-independent method

In this approach of speaker verification there is no constraint on the utterance text when the system is in use. So it must be trained to be able to cope with utterances of any text. The verification system does not know the text spoken by the speaker. The text could be user selected phrase or conversational speech [10]. Text-independent speaker verification system is used for applications with less control over user input and where the linguistic content of the speech is unknown [10].

2.3.4 Speaker Verification Methods Summary

When the linguistic content of the message is known, text-dependent and text-prompted speaker verification systems generally perform better than text-independent systems because they can model the characteristics of the specific phonetic-content contained in the speech signal [10, 11]. This thesis research work uses text-prompted method of speaker verification.

2.4 Components of a speaker verification system

Speaker verification system framework has three main process components; feature extraction, speaker modelling, and verification (decision making) [12].

2.4.1 Speaker Features Extractor

The feature extraction process component involves acquiring speech data, digitizing it into waveform, dividing it into frames and finally computing feature vectors for each frame [12]. The feature vectors contain speaker dependent characteristics.

2.4.2 Speaker Features Trainer

The speaker modelling process component generates a unique voiceprint (model) for each speaker and records (enrols) it into the database. This is where the speaker verification system learns the target client speaker.

2.4.3 Speaker Features Classifier

The verification (decision making) process component is a binary decision making process where the claimed speaker is either rejected or accepted but not both. It performs feature extraction from the newly coming test utterance, and then it matches the pattern of the new feature vectors of the claimed speaker with the already enrolled voiceprint. Then it compares the similarity measure result *x* to a threshold value *theta*. Finally it decides to accept (may be when *x* > *theta*) or reject (*x* <= *theta*) the trial of the claimed speaker using some decision theory [12, 13].

2.4.4 Speaker Verification System Components Summary

During training (enrolment), the system utilizes feature extraction and speaker modelling components. In the verification (testing) phase, it does feature extraction, feature matching and decision making. Figure 2.2 illustrates a generic speaker verification model comprising the three components in relation to the enrolment and verification phases.

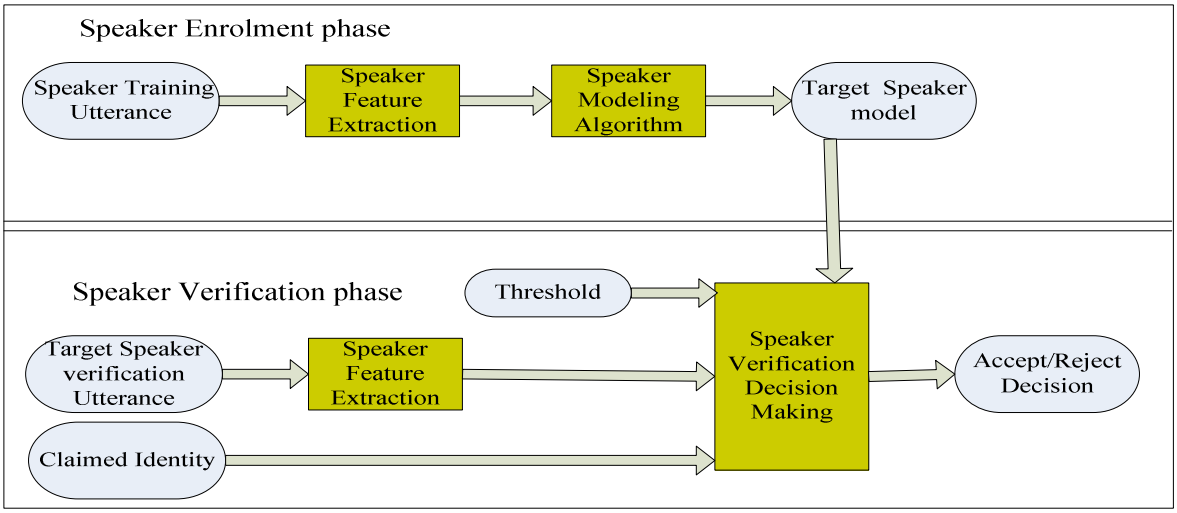

Figure 2.2: Components of a generic Speaker verification model.

2.5 Speaker dependent features

A spoken utterance consists of speaker characteristics, spoken phrases, emotions and noises [14]. Speaker dependent characteristics are correlated with the physiological and behavioral characteristics that exist in the voice source and vocal tract of the speaker [14].

From the viewpoint of their physical interpretation, speaker dependent features can be divided into short-term spectral features, voice source features, spectro-temporal features, prosodic features, and high-level features [14].

2.5.1 Short-term spectral features

Short-term spectral features are descriptors of the short-term spectral envelope (about 20-30 milliseconds in duration) as well as the resonance properties of the vocal tract [14, 15]. The speech signal continuously changes due to articulatory movements, and therefore, the signal must be broken down in to short frames of about 20-30 milliseconds in duration. Within this interval, the signal is assumed to remain stationary and a spectral feature vector is extracted from each frame [7, 14, 15].

2.5.2 Voice source features

These features characterize the glottal excitation signal of voiced sounds such as glottal pulse shape and fundamental frequency [14]. Fundamental frequency is the rate of vocal fold vibration. Other parameters are related to the shape of the glottal pulse, such as the degree of vocal fold opening and the duration of the closing phase [14]. The glottal voice source features are not as discriminative as vocal tract features. The amount of training and testing data for the voice source features can be significantly less compared to the amount of data needed for the vocal tract features [14]. A possible explanation for this is that vocal tract features depend on the phonetic content and thus require sufficient phonetic coverage for both the training and test utterances. But voice source features depend much less on phonetic factors [14].

2.5.3 Spectro-temporal features

Useful speaker-specific information is contained in the spectro-temporal signal details such as formant transitions and energy modulations [14]. A common way to incorporate some temporal information to features is through first and second order time derivative estimates [7, 14]. They are computed as the time differences between the adjacent vectors feature coefficients and usually appended with the base coefficients on the frame level. Spectro-temporal features span over tens or hundreds of milliseconds of speech duration [14].

2.5.4 Prosodic features

Prosodic features are the non-segmental long-term aspects of speech in hundreds of milliseconds, including for instance syllable stress, intonation patterns, language background, sentence type, speaking rate and rhythm [7, 9, 14]. They are supra-segmental since they normally extend over more than one phoneme segment [7]. They are created by certain special manipulations of the speech production system during the normal sequence of phoneme production. These manipulations could be based on subtle changes in the speech breathing muscles and vocal folds and/or via the movements of the upper articulators [7].

There are three main classes of prosodic features [9]; pitch, energy, and duration. Pitch features describe the intonation of sentences. The general trend for statements is downward pitch contour from the beginning to the end of the sentence. When the same sentences are spoken as questions, the pitch contour generally rises evenly from the beginning to the end of the sentence [7]. Energy features describe the intensity of the uttered words. Duration features represent the speed of talking and the number of pauses.

2.5.5 High-level features

High-level features attempt to describe conversation-level characteristics of speakers, such as characteristic use of words [14]. Speakers differ not only in their voice timbre and accent/pronunciation, but also in their lexicon - the kind of words the

speakers tend to use in their conversations. This includes the speaker's characteristic vocabulary (idiolect) [7, 14]. The idea in high-level features is to convert each utterance into a sequence of tokens where the co-occurrence patterns of tokens characterize speaker differences [14].

2.5.6 Speaker dependent features summary

There is to date no universally optimal feature set for use in speaker verification. But such a feature set [14] has to contain rich speaker dependent information that enables a classifier to more easily distinguish between different speakers. And also it has to be reliably obtainable and computationally feasible, even in situations where data is sparse.

2.6 Speaker Features Extraction

A speaker verification system then has to extract speaker dependent characteristics from others so as to reduce the dimensionality of the measurement space and to develop an enrolment model for the speaker [14]. Reducing the dimensionality of the measurement space is to mean that the data to be extracted is reduced by decomposing the original acquired speech data into sub-sets where each sub-set is in turn represented as single component in the final feature set.

The adequacy of a feature set for speaker verification lies in its ability to model signal properties that are unique for each speaker. The main speaker feature extraction techniques include Mel-Frequency Cepstral Coefficients (MFCCs), Linear Prediction Cepstral Coefficients (LPCCs) and Perceptual Linear Prediction Cepstral Coefficients (PLPCCs) [7, 9, 14].

2.6.1 Speech Signal Frequency Analysis

A feature extraction method involves speech signal frequency analysis to extract speaker feature vectors [14]. The reason is that many speech voice characteristics are difficult to measure explicitly so that they are captured implicitly by signal measurements [2]. In addition, the data rate of a waveform is extremely high.

Performing a frequency analysis provides information about the spectral content of the waveform at a significantly lower data rate [12].

To this end the cepstrum is applied [7] to the speech wave in equation (1). The cepstrum is defined as [7, 9]:

$$c_y\ (n) = G^{-1}(\log |G(y(n))|) \qquad (2)$$

where, G is the Discrete Time Fourier Transform and G^{-1} is the Inverse Discrete Time Fourier Transform. By moving the signal to the frequency domain G(y(n)), the convolution becomes a multiplication [7, 9]:

$$Y(w) = X(w)\ H(w) \qquad (3)$$

Further, by taking the logarithm of the spectral magnitude the multiplication becomes an addition:

$$\log|Y(w)| = \log|X(w)\ H(w)| \qquad (4)$$

$$\log|Y(w)| = \log|X(w)| + \log|H(w)| \qquad (5)$$

$$c_Y(w) = c_X\ (w) + c_H\ (w) \qquad (6)$$

The Inverse Fourier Transform $G^{-1}$is linear and therefore it works individually on the two components [7]:

$$c_y\ (n) = G^{-1}\ (c_X\ (w) + c_H\ (w)) \qquad (7)$$

$$c_y\ (n) = G^{-1}(c_X\ (w)) + G^{-1}(c_H\ (w)) \qquad (8)$$

$$c_y\ (n) = c_X\ (n) + c_H\ (n) \qquad (9)$$

In this IDTFT,G^{-1}, the slowly varying part $c_H(n)$ yields a cepstral component at low frequencies and represents an approximation to the cepstrum of the vocal system impulse response [7]. The component with fast variations $c_X(n)$ results in a cepstral component at high frequencies and corresponds to the cepstrum of the excitation [7].

2.6.2 Mel Frequency Cepstral Coefficients (MFCCs)

A “mel” is a unit of special measure or scale of perceived pitch of a tone [16]. The name ‘mel’ comes from the word ‘melody’ to indicate that the scale is based on pitch

comparisons. Mel-Scale cepstral coefficients are used to mimic human auditory processing [16]. Empirical studies have shown that the human auditory system resolves frequencies non-linearly, and the non-linear resolution can be approximated using the Mel-scale which is given by

$$M(f) = 1127.01048 \log_e f \qquad (10)$$

where f is a frequency [17]. Mel-frequency scale is linear until 1 KHz and logarithmic thereafter [17].

In speech processing, the mel-frequency cepstrum (MFC) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear Mel scale of frequency [16, 18]. Mel-frequency cepstrum coefficients (MFCC) are well known features used to describe speech signal [7, 16]. MFCCs and their time derivatives are coefficients that are computed at fixed frame rate and they collectively make up an MFC [7, 18]. MFCCs are based on the known evidence that the information carried by low-frequency components of the speech signal is phonetically more important for humans than carried by high-frequency components [16, 18].

The difference between the cepstrum and the mel-frequency cepstrum is that in the MFC, the frequency bands are equally spaced on the Mel scale, which approximates the human auditory system's response more closely than the linearly-spaced frequency bands used in the normal cepstrum [16, 18]. This frequency warping can allow for better representation of sound [18].

2.6.3 Linear Prediction Cepstral Coefficients (LPCCs)

In MFCCs, the Discrete Fourier Transform (DFT) or the Fast Fourier transform (FFT) decompose a signal into its frequency components. Linear prediction (LP) is an alternative spectrum estimation method to DFT or FFT. LP has good intuitive interpretation both in time-domain and frequency-domain [14].

In time domain [14], LP predictor equation approximates current sample signal $\hat{S}(n)$ as a linear combination of the past *p* observed sample signals S(n-k) as:

$$\hat{S}(n) \approx \sum_{k=1}^{p} a_k S(n-k) \quad (12)$$

where a_k are the predictor coefficients with k=1, 2, 3, ...,*p* and *p* is called the LPC analysis order. The prediction error signal, or residual, is defined as

e(n) = S(n)- $\hat{S}$(n). The coefficients a_k are usually determined by minimizing the residual energy e(n) and solving the resultant set of *p* simultaneous equations.

In frequency domain, the LP spectral model is defined as [14]:

$$H(z) = \frac{1}{1-\sum_{k=1}^{p} a_k z^{-k}} \quad (13)$$

and it consists of spectral peaks or poles only.

The output of the LPC analysis on a window of speech is the vector of *p* coefficients $a_1, a_2, .., a_k, .., a_p$ that specify the spectrum of an all-pole model that best matches the spectrum of frequencies inside the window. The predictor coefficients a_k themselves are used as features but they are usually transformed into robust and less correlated features called linear predictive cepstral coefficients (LPCCs). LPCCs are proved to be one of the most effective representation of speech signal for speaker verification [9,14]. The relationship between cepstrum coefficients c_n and prediction coefficients a_k is represented in the following equations [20]:

$$c_1 = a_1 \quad (14)$$

$$c_n = \sum_{k=1}^{n-1}\left(1-\frac{k}{n}\right) a_k\, c_{n-k} + a_n; \quad 1 < n \le p \quad (15)$$

$$c_n = \sum_{k=1}^{n-1}\left(1-\frac{k}{n}\right) a_k\, c_{n-k}; \quad n > p \quad (16)$$

It is usually said that the cepstrum, derived in such a way represents the "smoothed" version of the spectrum with much of the influence of the excitation removed [9, 20]. However, there is less freedom to apply non-linear processing to combat noise than there is with mel-cepstrum and also, LPC inherently gives uniform weighting to low- and high-frequency regions of the spectrum [9]. A non-linear frequency scale can be incorporated, but complicates the analysis to a greater extent than when using mel-cepstrum methods [9].

2.6.4 Perceptual Linear prediction Cepstral coefficients (PLPCCs)

Perceptual linear prediction (PLP) combines LPC analysis with psychophysics knowledge of the human auditory system so as to incorporate the non-linear frequency scale [9, 14, 15]. It includes perceptual aspects to the verifier and is more robust than the linear prediction cepstral coefficients (LPCCs) [14]. With respect to LPC, PLP analysis applies three transformations to the speech signal prior to building the all-pole model to simulate the perceptual properties of the human ear [9, 14]. These three psychophysics based transformations are critical band analysis, equal-loudness pre-emphasis and intensity-to-loudness conversion.

The critical band analysis simulates the non-uniform frequency resolution of the auditory system. The human ear has a higher frequency resolution at low frequencies than it does at high frequencies. This is achieved by mapping the frequency scale onto the Bark scale. The Bark frequency scale is a frequency scale on which equal distances correspond with perceptually equal distances. One bark is width of one critical band. Bark scale is linear until 500 HZ and logarithmic thereafter [9, 14].

Equal-loudness pre-emphasis simulates the non-equal sensitivity of hearing at different frequencies. Finally, intensity-to-loudness conversion simulates the non-linear relationship between the amplitude of a sound and its perceived loudness using a cube root amplitude compression.

The PLPCC derivation roughly involves the following steps [9, 14]:

1. Speech wave acquisition: f(n)
2. Spectral analysis: $\|DFT(f(n))\|^2$
3. Critical band analysis: frequencies mapped to Bark Scale
4. Equal-Loudness pre-emphasis: pre-emphasized by the simulated equal-loudness curve
5. Intensity-to-Loudness conversion: cubic root amplitude compression
6. Linear prediction analysis: IDFT, solving autoregressive coefficients, build all-pole model
7. Computing LPCC.

Apart from the use of LPC to achieve spectral smoothing, PLP analysis is very similar to MFCC analysis, but with perceptual properties incorporated in a way that is more directly related to psychophysical results as shown in Table 2.1 for a comparison of the two methods [9]. In recent years a number of recognition systems have used PLP-based cepstral coefficients as acoustic features, and experimental evidence suggests that overall they give performance that is comparable with that obtained using MFCCs [9, 15].

Table 2.1: Comparison of properties of MFCCs and PLPCCs

MFCCs	PLPCCs
Cepstrum-based spectral smoothing	LPC-based spectral smoothing
pre-emphasis applied to spectrum	Equal-loudness pre-emphasis applied to spectrum
Triangular mel-filters	Critical-band analysis
Logarithmic amplitude compression	Cube root amplitude compression

2.6.5 Dynamic Cepstral Coefficients

Cepstral coefficients do not capture temporal information. For this purpose, Delta-cepstral and Delta-Delta-Cepstral can be computed from the static cepstra [14].

2.6.6 Speaker Final Feature Vectors

A single feature vector consists of cepstral, delta-cepstral and delta-delta-cepstral coefficients [9, 14, 19, 20]. The cepstral can be any of MFCC, LPCC, or PLPCC. It also consists of norm energy. Thus the dimension of each feature vector is determined by the total number of the coefficients [14]. Then a defined number of feature vectors per second need to be extracted from the speech signal (waveform). This final feature vector characterizes the spectral shape of the signal for each time slice [9, 14].

2.6.7 Speaker Features extraction summary

MFCC, LPCC and PLPCC described above are well known techniques used in speaker verification to describe signal characteristics, relative to the speaker

discriminative vocal tract properties [9, 14, 15, 19]. Each method results in the cepstrum coefficients, but the method of computation differs. Cepstrum representation of the speech signal has shown to be useful in practice. However, it is quite sensitive to the environment and noise [21]. Noise at one frequency destroys the shape and degrades performance. Therefore, in practice speech signal is usually pre-processed to achieve more precise representation. This process usually includes noise removal and pre-emphasis [9, 21, 22, 23].

Experimental evaluation of recognition accuracy of the MFCC, LPCC and PLPCC is made in [22] and result of this report is that MFCC slightly outperforms other types [22].

MFCCs are based on the filtering of spectrum using properties of human speech perception mechanism. On the other hand, LPCCs are based on the autocorrelation of the speech frame and the main drawback of LPCCs is that it does not resolve the vocal tract characteristics from the glottal dynamics [21], which vary from person to person and might be useful in speaker verification. And, it is generally considered that LPCCs are computationally less expensive. The all-pole model used in the linear predictive coding analysis provides a good model for the voiced regions of speech and quite bad for unvoiced and transient regions [24].

Each feature extraction method models each time-slice of the speech signal as a single spectral shape [18, 19]. The extracted speaker feature sets will be used as inputs to the different speaker modelling/classification techniques.

2.7 Speaker Modelling and Features Matching

Speaker features extraction strips off unnecessary information from the speech signal and hence reducing its data rate while converting the properties of the signal which are dependent only on the speaker and important for the pattern recognition task to a format that simplifies the whole verification process. These feature sets are used to parametrically represent (model) the speaker for the upcoming tasks of the speaker verification process.

Once the speaker feature vectors are extracted from the speaker's training utterances, a model of the speaker must be created, trained and stored into the speaker verification system database. Speaker modelling is a process of enrolling the speaker to the verification system by constructing a model of his/her voice, based on the features extracted from his/her speech sample [23]. It is done using the enrolment (training or learning) mode of the speaker verification system. The speaker model is used to do feature matching with the feature vectors extracted from test utterances.

2.7.1 Speaker Features Matching

Speaker features matching is a process of computing a matching score so as to measure the similarity between the speaker features extracted from the unknown test speech sample and an existing speaker model [23]. It is done in the verification phase of the system. The speaker features matching scores are processed using hypothesis testing to obtain the final decision.

2.8 Speaker Modelling Techniques

In speaker verification, speaker models can be divided into template (nonparametric) models and stochastic (parametric) models [23]. In template models, training and test feature vectors are directly compared with each other with the assumption that either one is an imperfect replica of the other. The amount of distortion between them represents their degree of similarity. Dynamic time warping (DTW) [25] and Vector quantization (VQ) [26] are representative examples of template models for text-dependent and text-independent speaker verification methods, respectively.

In stochastic models, each speaker is modelled as a probabilistic source with an unknown but fixed probability density function. The training phase is to estimate the parameters of the probability density function from a training sample. Matching is usually done by evaluating the likelihood of the test utterance with respect to the model. The Hidden Markov Model (HMM) [26] and the Gaussian Mixture Model

(GMM) [27] are the most popular stochastic models for text-dependent and text-independent verification methods, respectively.

According to the training paradigm, speaker models can also be classified into generative and discriminative models [12]. The generative models such as GMM and VQ estimate the feature distribution within each speaker. The discriminative models such as artificial neural networks (ANN) [28] and support vector machines (SVM) [12, 29], in contrast, model the boundary between speakers.

In summary, a speaker is characterized by a speaker model such as VQ, GMM or SVM. At run-time, an unknown speech voice is first represented by a collection of feature vectors, then evaluated against the target speaker models recoded during enrolment. The speaker models so far can be summarized as in Table 2.2.

Table 2.2: Summary of speaker models

Model	Template	Stochastic	Text-dependent	Text-independent	Generative	Discriminative
DTW	√		√		√	
VQ	√			√	√	
HMM		√	√		√	
GMM		√		√	√	
ANN		√		√		√
SVM		√		√		√

2.8.1 Dynamic Time Warping

Dynamic Time Warping (DTW) [2, 30] is a process by which each utterance is represented as a sequence of short-term spectral vectors for text-dependent speaker verification. The trial-to-trial timing variation of utterances of the same text is normalized by aligning the analyzed feature vector sequence of a test utterance to the template feature vector sequence using a DTW algorithm. The overall distance between the test utterance and the template is used for verification decision.

2.8.2 Vector Quantization

Vector quantization (VQ) [9, 14] also called centroid model, is a process of mapping vectors of a large vector space to a finite number of regions applying a

clustering algorithm in that space so that to group similar vectors together. Each region is called a cluster and is represented by its centre (called a centroid). A collection of all the centroids makes up a codebook. The amount of data is significantly less, since the number of centroids is smaller than the number of vectors in the original sample. This will reduce the amount of computations needed for comparison in next stages. Even though the codebook is smaller than the original sample, it still accurately represents a person's voice characteristics [9, 14].

It is one of the simplest speaker models for text-independent speaker verification method. In the following, we denote the test utterance feature vectors by X = $\{x_1,\ldots, x_T\}$ and the reference vectors by R = $\{r_1,\ldots, r_k\}$. The average quantization distortion is defined as [14],

$$D_Q(X, R) = \frac{1}{T}\sum_{t=1}^{T} \min_{1\le k\le K} d\left(x_t, r_k\right) \qquad (17)$$

where $d(.,.)$ is a distance measure such as the Euclidean distance $||x_t - r_k||$. A smaller value of D_Q indicates higher likelihood for X and R originating from the same speaker. D_Q is not symmetric as $D_Q(X, R) \neq D_Q(R, X)$ [14].

2.8.3 Hidden Markov Model

Hidden Markov Model (HMM) [2, 30] efficiently models the statistical variation in spectral features of a speaker for text-dependent speaker verification method and with better accuracy than DTW. It models a speaker by characterizing the utterances as sequences of subword units. The subword could be phonetic transcriptions of spoken utterances or acoustic segments extracted directly from the acoustic signal without using any linguistic knowledge [2].

HMMs [30] are very popular for sequence modelling so as to represent different states of a system. They are finite state models and the system can occur in only one of those states at any given moment. They model a stochastic process defined by a set of states and transition probabilities between those states, where each state describes a stationary stochastic process and the transition from one state to another state describes how the process changes its characteristics in time.

Each state of the HMM models the generation of the observed symbols or feature vectors using a stationary stochastic emission process [30]. For a given observation of a feature vector however, the state in which the model has been when emitting that vector is not known. The underlying stochastic process is therefore hidden from the observer. Each state of the HMM models a certain segment of the vector sequence of the utterance. These segments (which are assumed to be stationary) are described by the stationary emission processes assigned to the states, while the dynamic changes of the vector sequence will be modelled by transitions between the states of the HMM.

The three basic parameters in HMMs are the initial state distribution, the state-transition probability matrix, and observation probability distribution [30]. Form of HMM depends on the speaker verification method [31]:

- Phrase models: for fixed-text
- Phoneme models: for text-prompted
- Single state HMM models: for text-independent
- Gaussian mixture models: for high number of mixture components

2.8.4 Gaussian Mixture Model

Gaussian mixture model (GMM) [32, 33] is a stochastic speaker modelling technique for text-independent speaker verification method. It can be considered as an extension of the VQ model, in which the clusters are overlapping. That is, a feature vector is not assigned to the nearest cluster as in VQ, but it has a nonzero probability of originating from each cluster. A GMM is composed of a finite mixture of multivariate Gaussian components. A GMM, denoted by λ, is characterized by its probability density function [14]:

$$p(X|\lambda)=\sum_{k=1}^{K} P_k N\left(X|\mu_{k,k}\right) \qquad (18)$$

where K is the number of Gaussian components, P_k is the prior probability (mixing weight) of the k^{th} Gaussian component, and

$$N\left(X|\mu_{k,k}\right) = (2\pi)^{\frac{-d}{2}} |_k|^{\frac{-1}{2}} \exp\{ \frac{-1}{2} (x-\mu_k)^{T} (\frac{X-\mu_k}{k})\} \qquad (19)$$

is the d-variate Gaussian density function with mean vector μ_k and covariance matrix $_k$. The prior probabilities $P_k \geq 0$ are constrained as$\sum_{k=1}^{K} P_k = 1$.

Training a GMM [14] consists of estimating the parameters $\lambda=\{P_k, \mu_{k,k}\}_{k=1}^{K}$ from a training sample $X = \{x_1,\ldots, x_T\}$. The basic approach is maximum likelihood (ML) estimation. The average log-likelihood of X with respect to model λ is defined as [14]:

$$LL_{avg}(X, \lambda) = \frac{1}{T} \sum_{t=1}^{T} \log \sum_{k=1}^{K} P_k N\left(x_t | \mu_{k,k}\right) \qquad (20)$$

The higher the LL_{avg} value, the higher the indication that the unknown vectors originate from the model λ.

In GMM based speaker verification, a speaker-independent world model or universal background model (UBM) is first trained [14]. This background model represents speaker-independent distribution of the feature vectors. When enrolling a new speaker to the system, the parameters of the background model are adapted to the feature distribution of the new speaker. The adapted model is then used as the model of that speaker so that its model parameters are not estimated from scratch rather from this prior knowledge [14]. It is possible to adapt all the parameters, or only some of them (e.g. only the mean vectors) from the background model [32].

In the verification (testing) mode, commonly referred to as Gaussian mixture model – universal background model (GMM-UBM), the match score depends on both the target model (λ_{target}) and the background model (λ_{UBM}) via the average log likelihood ratio [14]:

$$LLR_{avg}(X, \lambda_{target}, \lambda_{UBM}) = \frac{1}{T} \sum_{t=1}^{T} \{ \log p\left(x_t | \lambda_{target}\right) - \log p(x_t | \lambda_{UBM}) \} \qquad (21)$$

which essentially measures the difference of the target and background models in generating the observations $X = \{x_1,\ldots, x_T\}$. The use of a universal background model (UBM) for all speakers makes the match score ranges of different speakers comparable [14].

2.8.5 Artificial Neural Network

Artificial Neural Network (ANN) [28] is, like SVM, a discriminative speaker modelling technique for speaker verification. Instead of training of an individual model for each speaker, neural networks are trained to model differences among known speakers and therefore, require fewer amounts of parameters and more efficiently performs in training and verification phases. A potential advantage of ANN is that feature extraction and speaker modelling can be combined into a single network, enabling joint optimization of the speaker-dependent feature extractor and the speaker model [28].

Multi-Layered Perceptrons (MLPs) are one of the simplest types of neural networks, which are feed-forward networks and universal approximators [34]. An MLP consists of three or four layers [34]; an input layer, an output layer, and (an) intermediate or hidden layer(s). Processing elements (neurons) in the input layer only act as buffers for distributing the input signal to neurons in the hidden layer. Each neuron in the hidden layer sums up its input signals after weighting them with the strengths of the respective connections from the input layer and sends to the output layer. Training an MLP network consists of adjusting its weights using a training algorithm. The training set is collection of input-output patterns that are used to train the network. The testing set is a collection of input-output patterns that are used to assess network performance. The learning rate is a scalar parameter used to set the rate of adjustments.

Multi-layered networks are capable of performing just about any linear or nonlinear computation, and can approximate any reasonable function arbitrarily well [34]. Such networks overcome the problems associated with the perceptron and linear networks. However, while the network being trained may be theoretically capable of performing correctly, backpropagation and its variations may not always find a solution [34].

2.8.6 Support Vector Machine

Support vector machine (SVM) [12, 14, 29] is a discriminative speaker modelling technique for speaker verification. In speaker verification, one class consists of the target speaker training vectors (labelled as +1), and the other class consists of the training vectors from an impostor background population (labelled as -1). SVM classifies an input feature vector into either class +1 or class -1 [29].

2.9 Fusion

Fusion [14] is a technique where multiple sources of evidence are used for the purpose of verifying an individual speaker. One case is when a number of different feature sets are first extracted from the speech signal and then the same speaker modelling technique is applied for each feature set. Finally the sub-scores or decisions are combined. This implies that each speaker has multiple speaker models stored in the database. Another case of fusion is when the same feature set is modelled using different speaker modelling techniques and then the sub-scores are fused.

2.10 Speaker Features Normalization

Speaker features normalization techniques modify the spectral representation of incoming speech waveforms in an attempt to reduce mismatches between feature vectors of a speaker. The acoustic speech signal carries phonetic information, affective attributes, speaker characteristics and transmission path information [14]. But it is subject to many variations, most of which are undesirable. Speakers cannot repeat an utterance precisely the same way from trial to trial [2]. Any mismatch between the training and testing conditions dramatically decreases the accuracy of speaker verification [2]. It is known that voiced speech sounds (vowels, nasals) are more discriminative than fricative and stop sounds while short-duration utterances require special care [14]. But then addressing the mismatch starts from locating the speech segments which are the real objects of interest from the given audio signal, the process of which is called voice activity detection [14].

In order to compensate for the variations, normalization techniques have to be applied speaker features [2]. One typical speaker features normalization technique is the parameter domain (spectral) normalization where by cepstral coefficients are averaged over the duration of the entire utterance and, the averaged values are subtracted from the cepstral coefficients of each frame [2, 14]. This method is effective for text-dependent speaker verification method where longer utterances are used [2]. It reduces linear channel effects on delta-cepstral coefficients and long-term spectral variations [2].

A second normalization technique is for likelihood (similarity or distance) [2] values that use a likelihood ratio and the cohort (representative of the population near the claimed speaker) do not include the claimed speaker model. The likelihood ratio is the ratio of the conditional probability of the observed measurements of the utterance given the claimed identity is correct, to the conditional probability of the observed measurements given the speaker is an impostor (normalization term). A positive log-likelihood ratio indicates a valid claim, whereas a negative value indicates an impostor.

A third normalization method which is based on a posteriori probability [2] and the cohort model includes the claimed speaker model. The difference between the normalization method based on the likelihood ratio and that based on a posteriori probability is whether or not the claimed speaker is included in the impostor speaker feature set for normalization. The use of a single background model for calculating the normalization term has become the predominate approach used in speaker verification systems.

Another way of handling the mismatch is model-domain compensation which involves modifying the speaker model parameters instead of the feature vectors [14].

To correct some speaker-dependent offsets that are not compensated by feature normalization and model compensation, score normalization method [14] can be applied. In score normalization [14], the “raw” match score is normalized relative to a set of other speaker models known as cohort. The main purpose of score normalization

is to transform scores from different speakers into a similar range so that a common (speaker-independent) verification threshold can be used. Different possibilities are available for computing the imposter score distribution like Znorm, Hnorm, Tnorm, Htnorm, Cnorm and Dnorm [35].

In Znorm [35], each speaker model has different dynamic range of matching scores. In order to normalize the matching scores of different models, the model-specific mean and the variance of the matching score are computed, and these parameters are used to normalize the matching score to follow a standard normal distribution.

In Hnorm [35], the telephone speech is affected by the types of handset and microphone used. The effects of handset can be reduced by first recognizing which type of handsets is used for the input speech, and the matching score is normalized to follow a standard normal distribution using the handset-specific parameters mean and variance.

2.11 Speaker verification decision

In speaker verification, there are only two decision alternatives, accept or reject, regardless of the population size and therefore its performance approaches a constant [2]. So the decision is a 2-class hypothesis testing [12].

Suppose H_0 denotes that the speaker is an impostor, H_1denotes the speaker is indeed the claimed client speaker and S = $\{s_1 \dots s_N\}$ denotes an utterance consisting of a sequence of N frames. Each frame s_i corresponds to one feature vector. Let F(S) denote the score assigned to the whole utterance S by the speaker modelling technique. It is from this utterance score F(S) that a decision is made. To decide whether to authenticate a claimant, simply compare F(S) to a threshold, T. If F(S) is greater than T then authenticate the claimant else reject.

In the hypothesis testing there are four possible outcomes. Two outcomes are when the hypotheses are correct:

- H_0 is chosen when the claimant is an impostor

- H_1 is chosen when the claimant is the client.

The remaining outcomes are errors.

- A type I error occurs when the client is rejected (a false-rejection, FR).
- A type II error occurs when an impostor is authenticated (a false-acceptance, FA). For a very sensitive system, the type II error is preferred to be the lowest possible.

The threshold T may be set so that it minimises the number of errors made by the system, the rate of either FA or FR is at a fixed value and a desired FA to FR ratio is achieved. But it may be varied depending upon the application, for example, in high risk applications T may be set so that the chances of FA occurring is low. In speaker verification a common statistic to quote is the equal error rate (EER). This is the error rate that occurs when the threshold is set such that the rate of FA is equal to the rate of FR [12].

2.12 Corpora

A standard voice verification corpus is a requirement for a given speaker verification model so as to measure its performance achievement in comparison to other previous works. [14] describes that English NIST (National Institute of Standards and Technology) speaker verification corpus is used as one such standard. Some works use different corpora for training and testing. [12] used existing YOHO (English) corpus for testing and PolyVar (French) corpus for testing. In our work such a standard corpus is not available.

3 FEATURE AND MODEL SELECTION FOR SPEAKER VERIFICATION IN AMHARIC

3.1 Introduction

This chapter discusses about the feature vector selection along with the respective feature normalization processes, speaker model selection and the associated kernel functions applied in the Speaker Verification Model in Amharic (ATPSVM).

3.2 ATPSVM Text-Prompted method

The ATPSVM model uses text-prompted speaker verification method. As it is discussed in Section 2.3, for stronger security considerations text-prompted speaker verification method is more preferred to fixed-text or text-independent methods. When a speaker speaks its Speaker ID to the verification model, the system prompts Amharic text to be read by the claimed user. The speaker verification model takes the Amharic speech voice signal of the prompted text by that specific speaker as an input. Then, the Speaker ID and the Amharic speech voice of the prompted text are the inputs of the ATPSVM model.

3.3 Speaker Features Selection

In the ATPSVM model, 22-ordered Mel-Frequency Cepstral Coefficients (MFCCs) are applied to the Amharic speech voice signal of the specified text uttered by the speaker. It then extracts speaker dependent characteristics (feature vectors) as MFCCs. For this work, MFCCs are preferred to LPCCs or PLPCCs because in Section 2.5 it is shown that MFCCs can represent detail characteristics of individual speakers.

3.4 MFCCs Feature Vectors Extraction Process

Technique of computing MFCCs is based on the short-term analysis using about five steps to be described next, and thus from each frame an MFCC vector is computed [18]. Frame describes the change in the shape of the vocal tract within about10-20 ms in which case it is assumed to be constant within this duration [7, 9, 16,

18]. Each frame is represented by a single feature vector describing the average spectrum for a short time interval [9]. Figure 3.1 is a simple illustration for the extraction of MFCCs.

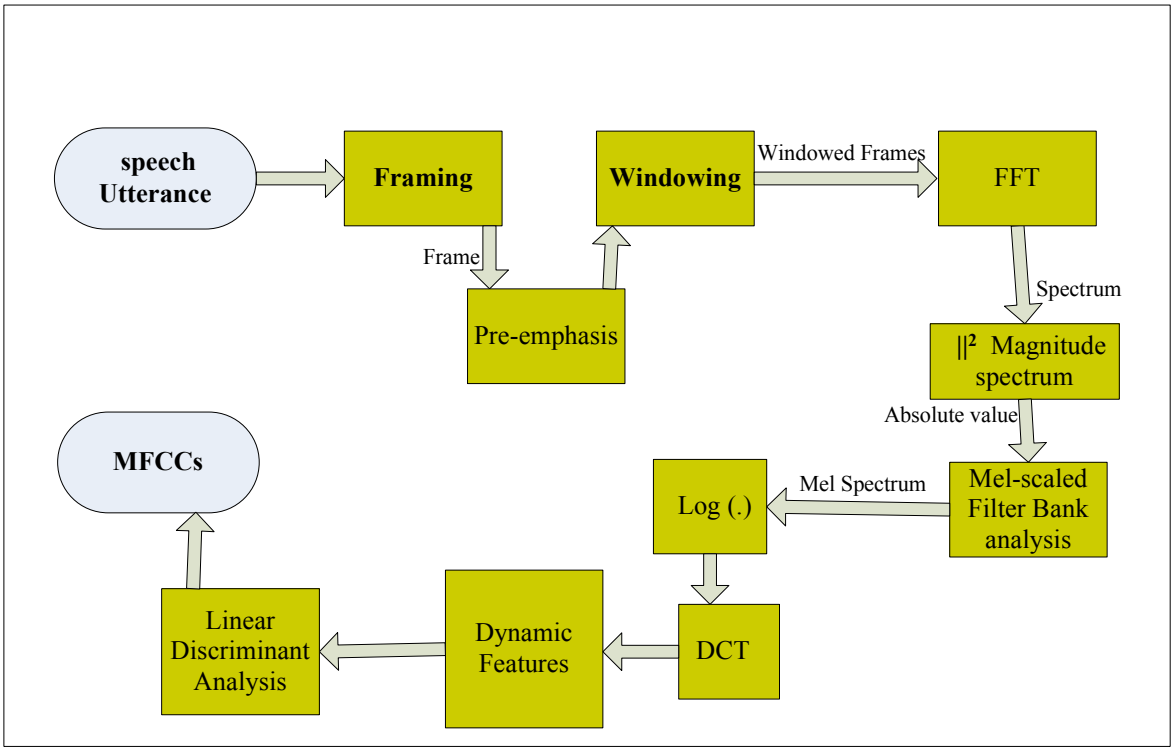

Figure 3.1 Illustration for extraction of MFCCs

Step1[Framing, pre-emphasis and windowing]: The first step, as shown in Figure 3.1, is dividing the input speech signal into blocks of short interval frames using overlapping smooth windows such as hamming [7, 16]. But before windowing, each frame is pre-emphasized so as to boost high frequency signal components by increasing the relative energy of high frequency spectrum [7]. It gives the higher formants in the vocal tract a better chance to influence the outcome. Speech signals contain two types of information, time and frequency [9]. In time space, sharp variations in signal amplitude are generally the most meaningful features [7, 9, 16]. In the frequency domain, although the dominant frequency channels of speech signals are located in the middle frequency region, different speakers may have different responses in all frequency regions [7, 9].

The hamming window function is the most popular windowing function in speaker verification and it is given by [19]:

$$W(n) = 0.54 - 0.46 \cos\left(\frac{2n\pi}{N-1}\right) \qquad (11)$$

where N is the size of the window or frame. This type of windowing is necessary to reduce any discontinuities at the edges of each frame, which would otherwise cause problems for the frequency analysis by introducing spurious high-frequency components into the spectrum [19].

Step2 [FFT]: The next step is taking the Discrete Time Fourier Transform (DTFT) or preferably Fast Fourier Transform (FFT) of the windowed signal. It transforms the discrete time-domain frame functions into frequency-domain periodic functions. One period of the frequency-domain function contains all the unique information [19].

Step 3[absolute value and mel-scaled filter bank analysis]: At this step, the absolute value of the DTFT of the windowed signal is calculated. Its output is the mel-scaled filter bank energy which enables to obtain the desired non-linear frequency resolution [9]. Normally filters are spread over the whole frequency range from zero up to the Nyquist frequency (which is half the sampling frequency) [7, 9]. However, band-limiting is often useful to reject unwanted frequencies or avoid allocating filters to frequency regions in which there is no useful signal energy [19].

Step 4[calculating the logarithm for the mel-scaled filter bank energies]: The fourth step is calculating the logarithm of the mel-scaled filter bank energies. The effect of inserting a transmission channel on the input speech is to multiply the speech spectrum by the channel transfer function [7, 9, 16]. But in calculating the logarithm, this multiplication becomes a simple addition which can be removed by subtracting the cepstral mean from all input vectors.

Step 5[DCT, dynamic features and linear discriminant analysis]: The last step is taking the Discrete Cosine transform (DCT) of the mel-scaled log-filter bank energies to get the MFCCs [9]. Then, we take the first and second derivatives of the cepstrals to get the other coefficients that represent the dynamic features of the speaker [7, 9].

Filter bank reduces the impact of excitation in the final feature sets [9]. The MFCCs are the amplitudes of the resulting spectrum [7, 9, 16]. The relevance of the Linear Discriminant Analysis is to provide a linear transformation of n-dimensional feature vectors (or samples) into m dimensional space (m < n), so that samples belonging to the same class are close together but samples from different classes are far apart from each other [16].

During an acoustic representation using MFCCs (mel cepstrum), the speaker characteristics are represented as feature vectors in the MFCCs. The MFCC features are based on the known variation of the human ear's critical bandwidths with frequency which is linear at low frequencies and logarithmic at high frequencies [17].

3.5 Speaker Features Normalization

The extracted MFCCs feature vectors are normalized using the parameter domain (spectral) normalization method so as to address mismatches between different utterances of a given speaker.

By this normalization technique, cepstral coefficients of the extracted MFCCs are averaged over the duration of the entire utterance and, the averaged values are subtracted from the cepstral coefficients of each frame. This method is effective for text-dependent speaker verification method as discussed in chapter 2.

Then the Min-Max normalization is applied to the final speaker feature vectors *s*:

$$\text{Min-Max: } s' = (s - \min) / (\max - \min) \qquad (33)$$

Equation (33) preserves the relationships among the original data values. Thus, the observation values under each feature vector are scaled such that the smallest value becomes zero and the largest value becomes one. That is, all values of the final feature vectors s' resulted from equation (33) will be in [0, 1].

3.6 Speaker Model Selection

As to modelling the speaker in the ATPSVM model, Support Vector Machine (SVM) is applied to the final normalized feature vectors. So that during the training

phase of the new system SVM creates and trains the speaker model. And during the testing phase, it classifies the final normalized feature vectors into either the impostor class (class -1) or the target client class (class +1). It makes the accept/reject decisions. The reason for using SVM is that speaker verification is basically a two-class (accept or reject) decision making process and, Section 2.8 describes that SVM algorithm is well suited for a two-class classification problem [45].

3.7 SVM Speaker Modelling Process

SVM is basically a computer algorithm that learns by example to assign labels to objects. It maximizes a particular mathematical function with respect to a given collection of data. SVM classifies a speaker using the feature vector X and discriminant function f [29]:

$$f(X)=\sum_{i=1}^{N} a_i t_i K(X, U_i) + d \qquad (22)$$

where U_i is the i^{th} support vector trained using training data with $t_i \in \{-1,+1\}$ as its label. The coefficient a_i is the corresponding weight for U_i with $a_i > 0$ and $\sum_{i=1}^{N} a_i t_i = 0$. The support vectors U_i, their corresponding weights a_i and the bias term *d* are determined from a training set using an optimization process [29]. K (. , .) is the kernel function.

3.8 SVM kernel functions

SVM Kernel function is the one that maps the training data into kernel space during SVM training.

The main SVM kernel functions are [46]:

a) Linear: Linear kernel or dot product.
b) Quadratic: Quadratic kernel.
c) RBF: Gaussian Radial Basis Function kernel.
d) Polynomial: Polynomial kernel.
e) MLP: Multilayer Perceptron kernel.

Only (a) from the list is linear SVM where as the remaining (b)-(e) are non-linear SVM.

3.8.1 Linear SVM

For the linear SVM, the kernel function K (. , .) is designed in such a way that it can be expressed as [29]

$$K(x; y) = \Phi(x)^T \Phi(y) \qquad (23)$$

where Φ(x) is a mapping from the input space to kernel feature space of high dimensionality. The kernel function allows computing inner products of two vectors in the kernel feature space [29].

Using the labelled training vectors, the linear SVM optimizer finds a separating hyperplane that maximizes the margin of separation between the two classes. As illustrated in Figure 3.2 [14], it models the decision boundary between the two classes as a separating hyperplane. And, in a high-dimensional space, the two classes are easier to separate with a hyperplane. The linear hyperplane in the high-dimensional kernel feature space corresponds to a nonlinear decision boundary in the original input space such as the MFCC space [14, 46, 47]. Thus, the linear SVM classifier is characterized as follows:

- The classifier is a separating hyperplane.
- The linear classifier relies on dot product between vectors so as to measure the distance between them.
- The most important training points are support vectors; they define the hyperplane.

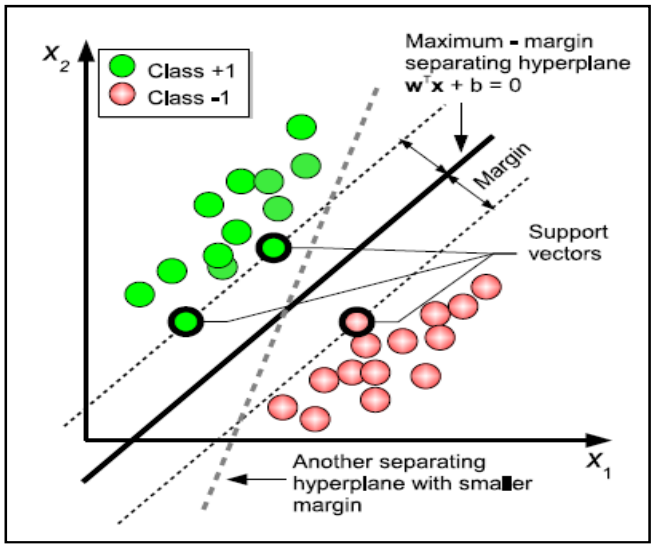

Figure 3.2 Illustration of Linear support vector machine

3.8.2 Non-Linear SVMs

In the non-linear SVM, the original input space is mapped to some higher-dimensional feature space where the training set is separable [47]. This could be illustrated as in Figure 3.3.

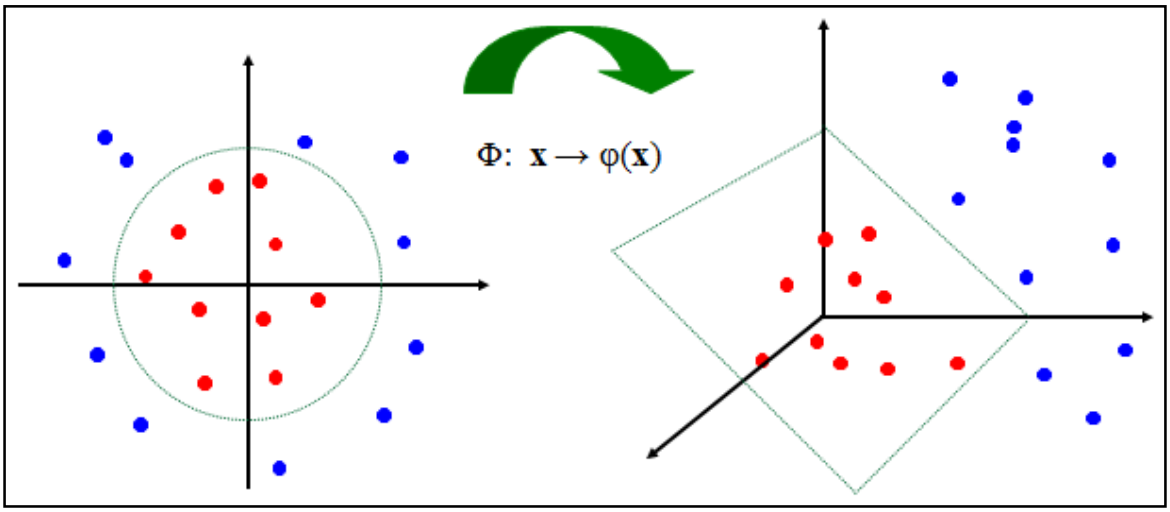

Figure 3.3 Illustration of the non-linear SVM

There are different non-linear SVM kernel functions. These include RBF, MLP and Polynomial.

I. Gaussian basis radial network kernel function (RBF) is defined as [47]

$$K(\mathbf{x_i}, \mathbf{x_j}) = \exp(-\frac{\left\|\mathbf{x_i} - \mathbf{x_j}\right\|^2}{2\sigma^2}) \qquad (52)$$

where σ is the distance between closest points with different classifications. x_i and x_j are data point vectors.

II. MultiLayer Perceptron (MLP) kernel function is defined as [47]

$$K(x_i,x_j) = \tanh(\beta_0 \mathbf{x_i}^T \mathbf{x_j} + \beta_1). \qquad (53)$$

Where *tanh* is the hyperbolic tangent function while β0 and β1 are constants

III. Polynomial of power *p* kernel function is defined as [47]

$$p\text{: } K(x_i,x_j) = (1 + x_i^T x_j)^p. \qquad (54)$$

Non-linear SVM locates a separating hyperplane in the feature space and classifies points in that space [47]. It does not need to represent the space explicitly; rather it defines kernel functions of the like listed above that play the role of the dot product in the feature space. Kernel functions are potentially very complex inner products in some feature Space. If kernel is given, no need to specify what features of the data are being used. Kernel methods are standard algorithms that exploit information about the inner products between data items [46, 47].

3.9 Speaker Features and Model Selection summary

It is discussed in this chapter that the ATPSVM model takes the speech voice of a prompted Amharic text as an input to verify a speaker. The model applies 22-order MFCCs to extract the feature vectors from the input speech signal. The detail of the MFCCs calculation process is outlined in this chapter. The model also applies the parameter domain (spectral) normalization followed by Min-Max normalization to the extracted feature vectors. Finally SVM is applied to the normalized feature vectors for modelling the speaker and for accepting or rejecting the speaker during verification. A set of kernel functions that are utilized in SVM are also discussed.

4 DESIGN OF THE ATPSVM MODEL

4.1 Introduction

In this chapter, the detail design of the desired Amharic Text-Prompted Speaker Verification Model (ATPSVM) is presented. It describes the architecture of the model and the algorithms applied. The training architecture and the testing architecture are separately presented. The training architecture of one-against-all is presented and its difference with the ATPSVM model one-against-each is also highlighted. The algorithms designed for the new model include feature extraction algorithm, speaker modelling algorithm and performance evaluation algorithm.

4.2 Architecture of the ATPSVM model for speaker training

Before using the speaker verification application, a speaker has to be enrolled in the system. This is done by uttering certain Amharic words and repeating each a number of times to the system. The enrolment phase enables the system to learn the speaker denoted by its Speaker ID and adds the respective speaker model to the Global speaker model set.

Figure 4.1 shows the training architecture for a single speaker applied in [45] as one-against-all SVM classification strategy. In [45], for each text T only one model is created for a speaker. This model is the model of the speaker against all other speakers. This single model of the speaker will be added to the global model set. So, for each text T the global model set consists of N models; one model for each speaker.

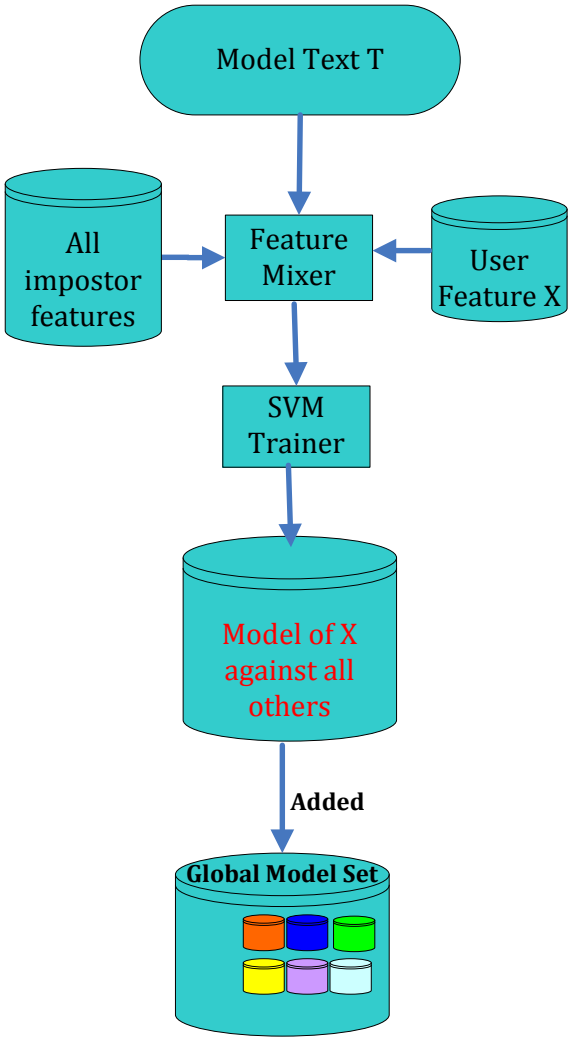

Figure 4.1 one-against-all training architecture

The detail description of each component in Figure 4.1 is as follows:

- Model Text T is the selected text the speech utterances of which are to be modelled.
- User feature X is the feature of the same model text uttered by the target speaker.
- All impostor features are all the features but not from the training speaker.
- Feature Mixer mixes all impostor features and target user feature X and determines the balance. At one instance, the feature vectors of all the N speakers are mixed. This is where the researcher believes unbalanced

feature training mix is created. Through this process the model in [45] trains the claimed speaker X and enables it to discriminatively classify the same.

- SVM Trainer does the learning and training process of the user feature X against all other impostor features.
- Model of X against all others is the output model of the training process for a single speaker.
- The Global Model Set consists of the voiceprints of all speakers. It consists of N speaker models; only one model for each speaker for each text T. The global model set is then used during the classification and decision making step of the speaker verification in [45] so as to feed the required model.

The proposed ATPSVM model on the other hand creates N-1 models for a single speaker from a single Amharic word T. It applies one-against-each strategy of SVM classification. This balances the test speaker feature vectors in the training set. These N-1 models will be added to the global model set. So for a single speaker, the global model set in the ATPSVM model of one-against-each consists of more models than the global model set in the one-against-all SVM strategy of [45]. The detail process of one-against-each SVM strategy applied in the ATPSVM model is depicted in Figure 4.2.

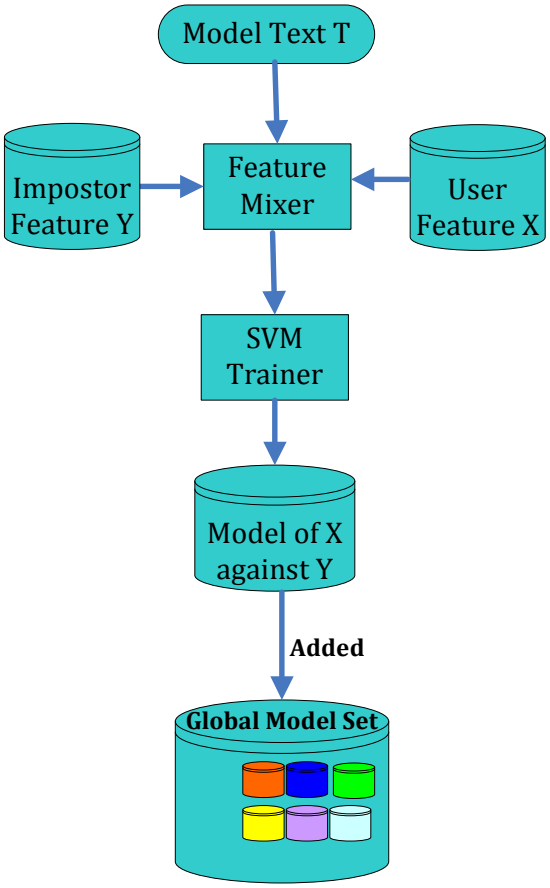

Figure 4.2 one-against-each Global Model Set development process

The detail description of some components in Figure 4.2 is as follows:

- Feature Mixer mixes impostor feature Y and target user feature X. Only the feature vectors of two speakers are mixed at one instant. This is where the researcher believes appropriate balance is created. Through this process the ATPSVM model trains and learns the claimed user and enables it to discriminatively classify the speaker X.
- SVM Trainer does the learning and training process of the user feature X against each impostor feature Y.
- Model of X against Y is created so that the ATPSVM model discriminating capacity with regard to user feature X gets increased. This is how a given speaker is uniquely identified from the rest of the speakers who are using the new model. Such N-1 models are created for a single speaker.

- The Global Model Set contains all voiceprints of each speaker X against each other feature Y. The global model set is then used during the classification and decision making step of the ATPSVM model.

4.3 Architecture of the ATPSVM model for speaker classification

A given utterance of an Amharic word has to pass through a set of steps so as to be used during the verification phase of the ATPSVM model. Normalized feature vectors are obtained by a similar step applied during the training phase of the ATPSVM model. During classification, what is new is that the SVM classifier classifies the final normalized feature vectors by using a previous speaker model of the speaker and comparing the score to a threshold. The classification process puts the speaker utterance feature vectors into either the impostor class or the target client class. The conclusion is accepting or rejection decision forwarded to the utterance feature vectors.

For a specific text T, in one-against-all SVM strategy model [45]:

- There is only one model to be fetched from the global model set
- Only one accept/reject classification step is applied.
- There is no aggregate metrics evaluation and analysis step.

The detail of the classification process for a single speaker using a single text in [45] is depicted in Figure 4.3.

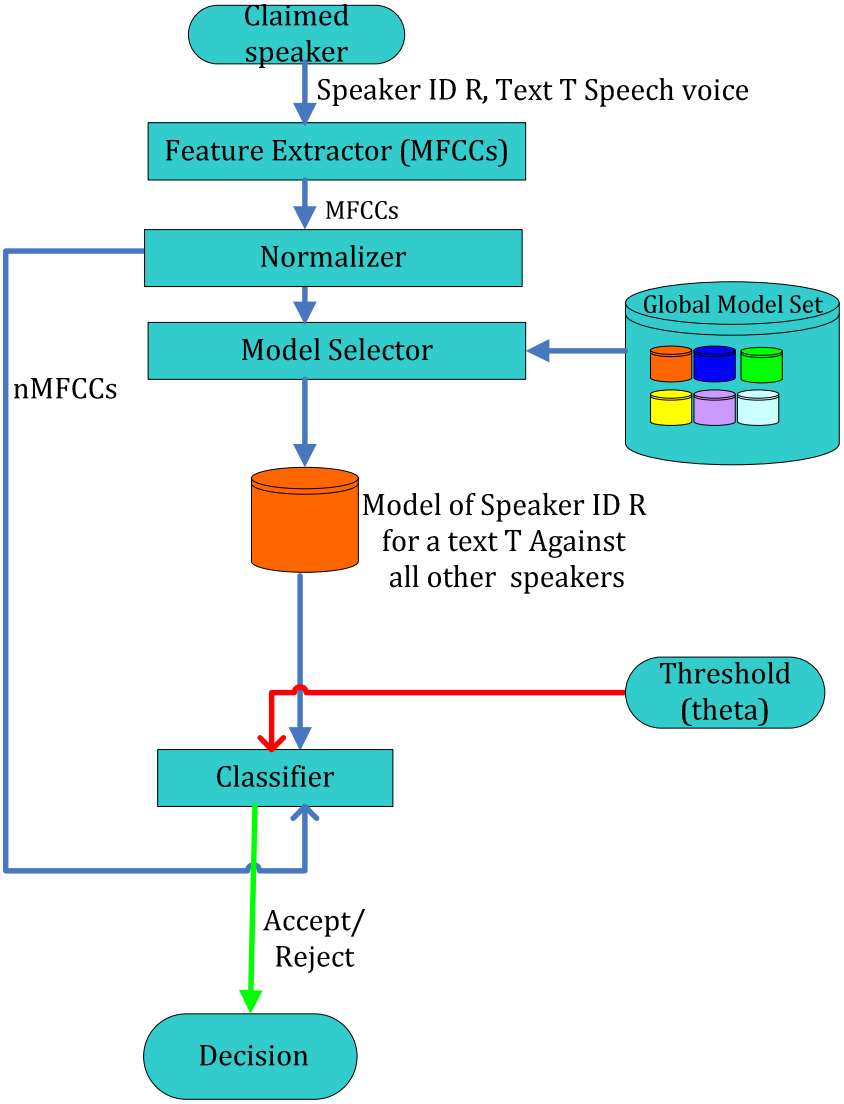

Figure 4.3: one-against-all speaker classification process

The description of the components in Figure 4.3 is as follows.

- Claimed speaker: The claimed submits the Speaker ID R & the given text test utterance to the model in [45].
- Feature Extractor: The MFCCs feature extractor extracts feature vectors from the utterance.
- Normalizer: The model in [45] then normalizes the feature vectors and produces normalized MFCCs (nMFCCs).

- Model Selector: The model selector retrieves the required single model of the claimed speaker against all other speakers for the text T from the Global Model Set.
- The Global Model Set: The Global Model Set consists of all the trained voiceprints (speaker models) available for the speaker verification model in [45].
- Classifier: For the single model of the claimed speaker retrieved from the global model set, a similarity measure is done by the classifier. The classifier takes in the nMFCCs and the threshold (theta) for the purpose of similarity measure and outputs either accept or eject decision.
- Decision: The decision at the text level is the final accept/reject decision result for the claimed speaker.

But in one-against-each SVM strategy model of the proposed ATPSVM model, the N-1 models in Global Model Set constructed from the claimed speaker for the specific Amharic word T are utilized. As shown in Figure 4.4 the final ACCEPT/REJECT decision is the aggregate of the N-1 accept/reject decisions made for each respective model in the global model set.

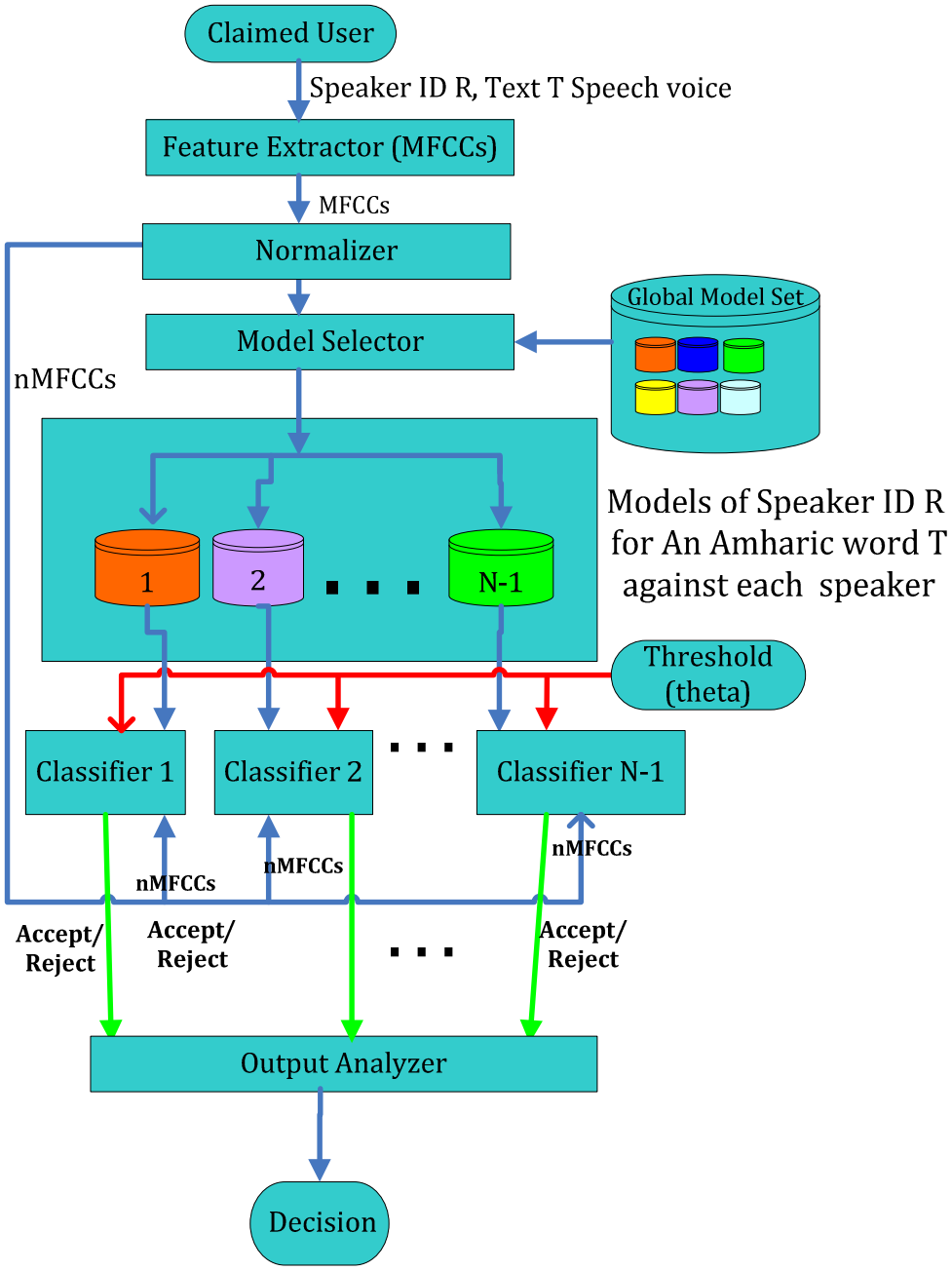

Figure 4.4: one-against-each classification architecture of the ATPSVM model .

The description of the architecture in Figure 4.4 is as follows.

- Normalizer: The ATPSVM model normalizes the feature vectors and produces normalized MFCCs (nMFCCs). It has also applied the min-max normalization to the feature vectors which is not done in [45]. Spectral normalization is done in both [45] & the ATPSVM.

- Model Selector: The model selector retrieves the required N-1 models from the Global Model Set. It retrieves all the N-1 models of Speaker ID R for the given Amharic word T against each other speaker of the same word.
- The Global Model Set: The Global Model Set consists of all the trained voiceprints (speaker models) available in the ATPSVM model.
- Classifiers: For each of the N-1 retrieved models a similarity measure is done by a respective classifier. Each of the respective N-1 classifiers takes in the nMFCCs and the threshold (theta) for the purpose of similarity measure and outputs either accept or eject decision. So that a set of N-1 accept/reject outputs will be available.
- Output Analyzer: The output analyzer takes in the entire N-1 accept/reject outputs from the N-1 classifiers. It counts TA, TR, FA, FR in the entire accept/reject values and reports as EER, Accuracy, Precision, and Recall at the Amharic word level.
- Decision: The decision based on EER and Accuracy at the Amharic word level is the final ACCEPT/REJECT decision result.

4.4 Functional Model Description

The claimed user submits its Speaker ID R to the speaker verification model. If the claimed Speaker ID is not found in the Global Model Set database, that specific speaker is not a registered user. This speaker is unknown by the ATPSVM model and hence it has to pass through the enrolment (training or learning) phase before using the speaker verification system. So the ATPSVM model starts training the model. Whether or not the speaker ID is registered, the model prompts text to be read by the respective user. This speaker utters the prompted text in Amharic. Then Amharic speech voice of the prompted text along with the claimed Speaker ID is used for training and verifying the speaker. To this end, from the uttered speech, MFCCs are calculated and normalized, and then SVM classification and decision making is applied to the final feature vectors.

Speaker verification decision is made by comparing each SVM classifier result with a certain threshold value used by the model. The threshold value of the ATPSVM model is determined by trial and error for many utterances of different Amharic speech texts from many speakers. The speaker is accepted only if its SVM classification value is greater than the threshold. Then the aggregate of accept/reject decisions for each N-1 model for an Amharic word speech from a speaker is used for the final ACCEPT/REJECT decision upon the speaker at the word level.

The ATPSVM model creates a set of impostor speaker models against a test utterance. Impostor models are fetched from the Global Model Set. The need for an impostor model is to normalize for utterance likelihood dependencies. The impostor model is speaker dependent. Impostor models serve as background models for each utterance of a speaker.

For each utterance *v* of an Amharic word *w* from a speaker *s*, the ATPSVM model creates an impostor model using the SVM training. The impostor model is built using speech utterances of *w* but not *v* of each speaker *s* on one-against-each manner. Each training model in this study comprises of a set of B background speaker models. These target (client) models include the B/2 nearest background speakers which are other utterances of *w* by the claimed speaker *s*. The remaining B/2 are impostor models generated from utterances of *w* by each impostor speaker. In this thesis work, it has been possible to generate 900 impostor models and another 900 client models from a single Amharic word using the 10 utterances from 10 speakers; the detail of which would be seen in the next chapter. Thus, each speaker enrolled in the ATPSVM system has a dedicated N-1 impostor models set with respect to each utterance of an Amharic word.

Finally, as a checking gate, speakers only accepted by the ATPSVM model get access to the desired service such as mobile banking or e-ticketing while rejected speakers are denied the service requested.

4.5 Algorithm design for the ATPSVM model

The proposed ATPSVM model prompts some Amharic words to be uttered repeatedly by the speaker. The ATPSVM model records separate speech wave file for each utterance and applies MFCCs feature extraction, feature normalization and SVM speaker classification. An utterance of a given word is iteratively taken for testing while the remaining utterances of the word are taken for training on Leave-One-Out basis. The pseudo code of the algorithms for feature extraction, normalization, classification and decision making as applied in the ATPSVM are presented hereafter.

4.5.1 Design of the Feature Extraction Algorithm for the ATPSVM model

The main steps involved in the feature extraction process can be grouped into two routines. Each routine handles subtasks of its own.

```
Routine one:
        For each utterance wave file;
                Calculate MFCCs;
                Normalize MFCCs' values;
                Save as file1.

Routine Two:
        For each Amharic_word ;
                For each test_utterance ;
                        Mix the features of remaining utterances of file1s;
                        Save as file2.
```

*file1 and file2 can be in Matlab or Ms Excel formats.

In routine one, MFCCs are extracted from each utterance wave file and then the feature vectors are normalized. The calculation process for MFCCs is described in Section 2.5. During normalization, the average value for the observations for each feature vector are calculated and deducted from each. This is what the parameter

domain (spectral) normalization does. Then the min-max normalization is applied so as to get the feature vector values in [0, 1].

The ATPSVM model is tried with different MFCCs orders such as 13 and 22. After applying the SVM speaker modelling technique, more accurate classification performance is obtained for 22-order MFCCs feature vectors.

In routine two, client and impostor models are generated for each utterance of each Amharic word.

4.5.2 Design of the Speaker Modelling Algorithm for the ATPSVM model

The SVM learns the claimed speaker from those 22-order MFCCs feature vectors and then classifies him/her into client or imposter. During the enrolment (training) phase, SVM creates a model structure for the speaker. Then during verification (testing or classification) phase, it classifies the speaker into either imposter or client group using the MFCCs feature vectors.

This work uses feature vectors from 1000 speech wave files and is designed in such a way that it will be tested using RBF, Polynomial, MLP and Linear kernel functions of SVM interchangeably. The performance the ATPSVM model is going to be reported for each of these Kernel functions.

As to learning, the speaker data are trained to an SVM client target value of +1 whereas the imposter data are trained to an SVM target value of -1. During verification, SVM generates a hyperplane that can separate speaker and imposter MFCCs features vectors.

The SVM based speaker modelling, classification and decision making algorithm for the ATPSVM model is presented in two routines.

```
Routine three:
    For each Amharic_word i;
        For each test_utterance j;
            Read file2 of i and j for training;.
            Read file1 of i and j for testing;
Routine four:
    For each file2;
        SVM train the MFCCs feature vectors;
        SVM classify the MFCCs feature vectors using
            file1and save the class as file3.
```

* *file1*and *file2* are outputs of the Feature extraction algorithm.

In routine three, respective training and testing feature vector files are loaded. In routine four, feature vectors are trained, classified and the classified data is reported.

4.5.3 Design of the decision making algorithm for the ATPSVM model

In the ATPSVM model, SVM classifies feature vectors into Class 1 (target or client) and Class 0 (impostor) by assigning each observation a value of 1 or 0 respectively. Please notice that for the client target class, class 1 or class +1 labels are interchangeably used. Similarly, class -1 or class 0 are interchangeably used to denote the impostor class. The ATPSVM model calculates the ratio *p* of count of class 1 observations flagged by the SVM classification. For the final speaker verification decision of the ATPSVM model, the ratio *p* is compared with the threshold *theta* for deciding whether to accept or reject the speaker. This algorithm is presented in two routines.

Routine five:

For each file3;
Calculate count performance P=#class1/(#class1+#class0);
Compare P with threshold theta; { P>theta or otherwise}
Increment by 1 only one of {TA, TR, FA, FR}

Output ← counts of TA, TR, FA, FR

Routine six:

Do aggregate metrics {EER, FAR, FRR, Accuracy, precision, recall}
*Do final Accept/Reject {*based on EER & Accuracy cut-off points*}*

**file3 is the* output class list of the SVM classification.

In routine five, the counts for TA, TR, FA and FR are obtained for each Amharic word. These values are taken as inputs for routine six so that the classified feature vectors further analyzed.

In routine 6, the aggregate metrics are calculated from the counts of TA, TR, FA, and FR for the final ACCEPT/REJECT decision. The aggregate metrics include FAR, FRR, EER, Accuracy, Precision and Recall. The final ACCEPT/REJECT decision is made based on EER and Accuracy. Higher accuracy and lower EER values are the desired values. The cut-off points for EER and Accuracy can be determined as per the security requirements of the service to which the ATPSVM model is to be applied as a security authentication mechanism.

4.6 Summary of the design for the ATPSVM model

In this chapter, the design and architecture of the ATPSVM model is presented. In addition, the MFCCs as speaker feature set extraction model and the SVM as speaker modelling technique are utilized in the model. Finally the algorithms applied in the design of the model are outlined. The next chapter, chapter 4, presents the top level implementation of the ATPSVM model

5 IMPLEMENTATION OF THE ATPSVM MODEL

5.1 Introduction

In this chapter, a description of the technical implementation of the ATPSVM model prototype is presented.

5.2 Format of Implemented Speech wave file

The Amharic speech wave files are recorded in a laboratory environment. Each wave file is created from a single utterance of a word. It is in Resource Interchange File Format (RIFF). RIFF is Microsoft's specification for the storage of multimedia files and hence suitably applicable in windows applications. A RIFF file starts out with a file header followed by a sequence of data chunks. The WAV file format is a subset of Microsoft's RIFF specification.

The speech wave files are recorded in 16000 Hz sampling rate. Sampling rate is the number of samples taken per second each of which is equal in size. Each sample has 16 bits size and hence a total of 1000 samples are generated for each wave file. It is the number of bits per sample to encode the data in the file. Only one channel is used and the maximum duration of each wave file is limited to one second. This optimizes the size and quality of the final feature vector set. Each speech wave file is saved to disk with *.wav* extension & as 32KB size. A total of 1000 such files are recorded as part of the implementation task of the ATPSVM model. The recording task is one tedious aspect observed during implementation.

5.3 Feature extraction implementation

Each Amharic speech wave file is captured as a set of frames. Each frame of data is a collection of sequential samples from the specified single channel combined into a single vector. Here each frame is set to incorporate 512 sequential samples. This is the window size or the frame size.

Frame rate or window shift is the number of frames that a speech acquisition device can capture per second. So, a frame rate of 40 Hz is applied and this is the respective window shift. Notice that the window size (frame size) and window shift

(frame rate) are independent. Normally, the window size will be larger than the frame rate so that successive windows overlap.

This enables the new model to capture enough feature vector data for a good discrimination of the owner of the utterance. Another important reason to set the frame size is that some devices require their inputs to be vectors of specific sizes. A pre-emphasis filter followed by the Hamming code is applied to the input speech wave file so as to reduce the error rates of the model. In general, in frame-based processing, all the samples in a frame are processed simultaneously. In sample-based processing, on the other hand, samples are processed one at a time. The advantage of frame-based processing is that it greatly increases the speed of the new model.

The number of MFCC coefficients used for the model is set to be 22. MFCCs give good discrimination and lend themselves to a number of manipulations. For instance it enables to apply the cepstral mean normalization so as to compensate the effect of microphones and audio channels. Only the first few 12-13 coefficients are used to represent a frame. But in this research work, by setting the number of coefficients to 22 an excellent result is obtained. Therefore MFCCs provide a low-dimensional, smoothed version of the log spectrum, and thus are a good and compact representation of the spectral shape.

A speaker ID column is added to the final normalized feature vectors of the MFCC file so that to identify the owner of the utterance. Thus each feature vector file will have 23 columns.

5.3.1 Sample feature vector output

A sample feature vectors output is given in Figure 5.1. It also shows the final normalized feature vectors.

Columns 1 through 11

-20.5755	-23.5868	-24.0295	-23.5978	-23.6776	-23.6383	-23.7000	-23.2783	-23.1540	-23.1875	-22.2399
-0.1003	-0.1608	-0.1624	-0.2997	-0.1543	-0.3751	-0.2662	-0.4325	-0.3661	-0.3139	0.9026
-0.0670	0.2653	0.5354	0.2676	0.4097	0.3788	0.3541	-0.0019	0.0394	-0.1580	0.9267
-0.1179	-0.2423	0.1131	0.0478	-0.0503	-0.1702	0.0624	0.0881	0.1539	0.2004	0.7645
-0.1194	-0.1148	-0.2613	-0.1288	-0.1131	0.0446	-0.1746	0.2031	0.0602	0.2932	0.5731
-0.0951	0.0657	-0.1304	-0.0305	-0.1236	0.0829	-0.2058	-0.0973	0.0256	-0.2125	-0.0669
-0.0268	0.1864	0.0900	-0.0150	-0.0502	-0.1834	0.0480	-0.0419	0.0328	-0.3904	0.0122
0.0061	0.0840	-0.1377	-0.0735	-0.1242	0.1841	-0.0542	0.0404	-0.1885	0.2428	0.3591
-0.0388	-0.0080	-0.0284	0.0966	0.1782	0.0707	0.1864	0.1882	-0.0781	0.1401	0.1748
-0.0267	-0.1165	0.2156	-0.0450	0.2111	0.0078	-0.1438	0.0869	0.2761	0.1870	0.0894
0.0216	-0.2664	0.0390	-0.1046	0.0152	-0.0284	-0.1185	-0.1204	0.1135	-0.1068	-0.1116
0.0115	0.0413	-0.0047	-0.1196	-0.1755	0.0284	0.1156	-0.1760	-0.0032	-0.1941	-0.1772
-0.0093	0.0164	-0.0362	-0.0625	0.0187	-0.0414	0.0918	-0.0455	-0.2626	0.2180	0.1611
0.0260	0.1465	-0.1141	0.1903	0.0149	-0.0429	0.1075	0.2859	-0.0673	0.2159	0.5248
0.0045	0.0696	-0.0848	0.0401	-0.1064	-0.0488	0.0633	0.0539	0.1300	-0.3104	0.0629
0.0079	0.0469	0.1128	0.1052	0.0710	0.0360	-0.0756	-0.0333	0.1290	-0.0291	-0.2345
0.0126	0.0409	-0.0370	0.0012	-0.0156	-0.0162	-0.0857	-0.0921	-0.2030	-0.0926	-0.1940
-0.0065	-0.0603	0.0324	-0.0433	-0.0626	0.0651	-0.0953	0.0727	0.0511	0.1381	0.1219
-0.0012	-0.0818	-0.0478	-0.2494	-0.0144	0.0674	-0.2424	-0.0733	-0.0680	0.0935	0.1957
0.0149	0.0988	-0.0511	0.0035	-0.0132	-0.0151	0.0577	-0.0005	0.0497	-0.0642	-0.0377
0.0130	-0.0407	-0.0600	0.1913	-0.0033	-0.0619	-0.0573	-0.1359	-0.0496	0.2571	-0.0199
0.0223	0.0517	0.0221	0.1193	0.1555	-0.0937	0.0143	0.0820	0.0212	0.0458	0.0255

Columns 12 through 22

-20.7521	-20.0927	-18.7448	-17.7271	-18.6815	-19.3874	-21.8476	-20.5780	-20.8028	-20.0900	-20.6474
-0.6715	-2.7295	-1.8835	0.2322	0.8271	0.1686	-2.0821	-2.6547	0.1923	0.3478	0.2864
1.1812	1.2572	1.6102	0.0132	-0.1053	0.8016	1.2649	0.5348	0.7942	0.2506	0.3008

After Normalizations &Speaker ID added

Columns 19 through 23

0.9866	0.9761	0.9893	0.9839	0.9862
0.9917	0.9837	0.9946	0.9836	0.9847
0.9923	0.9720	0.9924	0.9875	0.9886
0.9974	0.9731	0.9956	1.0000	1.0000
0.9827	0.9872	0.9968	0.9930	0.9928
0.9904	0.9742	0.9827	0.9883	0.9917
0.9881	0.9776	0.9763	0.9747	0.9837
0.9832	0.9663	0.9933	0.9907	0.9836
0.9883	0.9789	0.9858	0.9902	0.9985
0.9944	0.9707	0.9922	0.9839	0.9909
0.9828	0.9710	0.9914	0.9827	0.9916

Figure 5.1: The 22 order MFCCs extracted then normalized and Speaker ID appended.

Notice that here each row corresponds to an observation or replicate for the 22 features (being 23 with Speaker ID added), and each column corresponds to a feature vector or variable (only one among the 22 or 23).

Each feature vector file is stored in such a way that the Speaker, the text and the utterance number can be identified.

5.4 Training Implementation

By the processes so far the final feature vectors are obtained. The next step is to train the new model for them. The SVM is applied for training so that it generates the SVM Structure that will be utilized later for verification. Four different kernels of the SVM are utilized interchangeably for training the feature vectors. These are the linear, MLP, RBF and Polynomial kernel functions. Unfortunately, the linear kernel function does not converge. And hence it is not applicable for the ATPSVM model. For the remaining three, the new model is able to generate the desired SVM structures.

5.5 Verification Implementation

During the verification processes, the SVM structure generated during training is utilized to classify the feature vectors. Performance and accuracy differences are observed between the different kernels applied. Further analysis is done on the aggregate performance metrics results for the final ACCEPT/REJECT decision.

5.6 Implementation tools

[1]. Speech wave files are recorded using WaveSurfer 1.8.8.

[2]. The prototype is implemented using Matlab 7.6.

[3]. Microsoft Excel worksheet is used for storing feature vectors.

[4]. Microsoft office Visio is applied to sketch the figures.

[5]. MedCalc 11.5.1.0 is used to sketch graphs.

[6]. Microsoft Word is used for documentation.

5.7 Coding and Libraries in the implementation

The main coding & library components applied during the implementation of the ATPSVM model are described as follows.

[1]. Each Amharic word utterance speech wave file is loaded using the built-in Matlab function *wavread()*.

[2]. A Matlab source code *mfcc.m* is adopted and customized so as to extract 22 ordered MFCCs feature vectors.

[3]. Own code is applied for spectral and min-max normalizations, and then for adding respective Speaker ID to the final feature vectors sets.

[4]. For SVM training, the model applied the built-in Matlab function *svmtrain()* to the final training feature vectors so as to generate the SVM Structures. But not easy and a lot of own effort is done to work with the different kernel functions.

[5]. For SVM classification, the built-in Matlab function *svmtclassify()* is applied to the final testing feature vectors along with the respective SVM Structure generated in [4]. Still a lot of own effort is done so as to apply for the prepared testing data and to get meaningful result for the thesis work.

[6]. Own code is applied for the analysis of aggregate performance metrics for the new model. For instance, *Svmclassify* classifies each row of the testing feature vector data into class 1 or class 0. Then counting the number rows per class is own done. Further analysis on each count and deriving the values for main performance metrics for each Amharic word such as EER, FAR, FRR, Precision and Recall are own codes. This final task also involves testing the performance of the model for the different threshold *theta* values and reporting the results.

Remark: The detail of the performance evaluation of the ATPSVM model is the focus of the next Chapter, Chapter 6 where as the detail implementation coding of the thesis work is presented in Annexes1, 2, & 3.

5.8 Summary the implementation for the ATPSVM model

The main implementation tasks applied for the ATPSVM model are presented. This includes implementation aspects done towards Amharic speech wave file formats, generating final feature vector forms, training and verification codes implemented. It also gives an insight how aggregate performance is generated and reported.

6 EXPERIMENTATION AND EVALUATION OF THE ATPSVM MODEL

6.1 Introduction

This chapter presents how experimentation is conducted on the ATPSVM model. It discusses about the data corpus used. It also describes how performance of the model is evaluated. Finally the performance of the model is reported.

6.2 Data collection

Amharic Speech wave data is collected from 5 male and 5 female people in a laboratory environment where high performance Laptop and high quality *inTEx* headphone are used. All of the test speech signals are tried to be noisy-free. The Amharic names of the ten numerals are selected as in Table 6.1 to evaluate the performance of the ATPSVM model. The respective numbers are also used for coding the Amharic words for simplicity of the upcoming tasks of the research.

Table 6.1 Amharic words & respective codes selected for experimentation

Code	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
Word	አንድ	ሁለት	ሦስት	አራት	አምስት	ስድስት	ሰባት	ስምንት	ዘጠኝ	አስር

Each speaker utters each Amharic word ten times so as to overcome mood variances and the respective utterance wave file is recorded in a separate file. So that 100 wave files are recorded from a single speaker. Thus, a total of 1000 wave files are recorded from the 10 people. This is required to capture potential variability of speaker speech voice so that performance is invariant of the speaker mood and other factors.

6.3 Testing Methodology

Final feature vectors are used as testing and training data sets. Test data set is extracted by iteratively taking one utterance from the 10 utterances of an Amharic

word from a speaker against other 9 speakers one by one; with the remaining 9 utterances of the two speakers being used for training. It is leave-one-out cross-validation experimentation method. It leaves out one utterance for testing while using the remaining others for training. Leave-one-out makes best use of the data as it enables to generate more test and train data sets from existing speech wave files.

Thus for each Amharic word, the new model iteratively takes one utterance of each speaker against other 9 speakers for testing. The respective utterance of other speaker is taken as impostor data set for the same. The remaining respective 9 utterances from the two speakers are taken as training data sets. Section 6.3.1 presents the detail of it. Applying this testing methodology, 900 training data sets, 900 impostor testing data sets & 900 client testing data sets are generated from each Amharic word experimented. That is, 10 utterances of the word for testing X from 10 speakers X against 9 other speakers X 2 test data sets (one client & one impostor)=1,800 total test data sets. One training data set is used for one client testing data set and its respective impostor testing data set.

Therefore, the ATPSVM model has utilized a total of 18,000 test data sets and 9,000 training data sets. For the convenience while applying this method, from each Amharic word 10 different test data set files along with the respective 10 training data set files are generated from the captured 100 speech wave files and are saved as MS Excel spreadsheet to disk. So, a total of 100 training data set files and 100 testing dataset files are generated & saved to disk from the ten Amharic words experimented in the ATPSVM model.

6.3.1 One-against-each pseudo code

For each Amharic word experimented ……(10 words)
  for each utterance i…….(10 utterances)
    for each speaker K ……(10 speakers)
      for each registered Other Speaker O and **O≠ K** ...(9 other speakers)
        Take the i^{th} utterance of K and O for testing
        Take the rest utterances of K and O for training

- This gives us a total of 10X10X10X9=9000.
- So the ATPSVM model utilized 9000 training data sets, 9000 true Speaker testing data sets and 9000 impostor (other speaker) testing data sets.

6.4 Performance of the ATPSVM model

During testing a given speaker using a given Amharic word, each of the ten utterances is evaluated against other speakers one by one. That is, for a given utterance number, the respective training data set along with the client & impostor test data sets is taken against other speakers one by one for SVM testing and training.

For the purpose of testing, final feature vectors of each client or impostor testing data set are presented to the ATPSVM model. During SVM classification, the new model assigns flags either 1 or 0 to each row so as to be classified as class 1(target) or class 0 (impostor). Then, the row count of 1s (a) and the row count of 0s (b) are obtained for a given test data set T. So the SVM score (R) of T is obtained by:

$$R=a/(a+b) \qquad (64)$$

R is then compared to a threshold *theta*. T could be impostor or client data set.

Performance of the ATPSVM is measured in terms of True-Acceptance (TA), True-Rejection (TR), False-Acceptance (FA), and False-Rejection (FR).

The definition of each term is as given next:

[1]. True-Acceptance (TA): is the MFCCs feature vectors (rows) identified correctly as class 1.

[2]. True-Rejection (TR): is the MFCCs feature vectors (rows) identified correctly as class 0.

[3]. False-Rejection (FR): is the MFCCs feature vectors (rows) identified wrongly as class 0. It means the system fails at that MFCCs feature vector row to recognize an authorized person and rejects that person as an impostor. It is called type I error.

[4]. False-Acceptance (FA): is the MFCCs feature vectors (rows) identified wrongly as class 1. It is the instance of the ATPSVM model incorrectly verifying or identifying an unauthorized person at that specific MFCCs feature vector row. Also

referred to as a type II error, a False Acceptance typically is considered the most serious of biometric security errors as it gives unauthorized users access to systems that expressly are trying to keep them out. To this end, the ATPSVM model is trained in such a way that the False-Acceptance value becomes the lowest possible.

The SVM score (R) obtained from equation (64) is analyzed by the ATPSVM model as given in Table 6.2.

Table 6.2: SVM score analysis

Test data set (T)	SVM Score (R)	Effect
client	R >theta	TA=TA +1
client	R <= theta	FR=FR+1
Impostor	R > theta	FA=FA+1
impostor	R <=theta	TR=TR+1

For all Client data sets of an Amharic word: TA + TR + FA + FR =900

For all impostor data sets of an Amharic word: TA + TR + FA + FR =900

The ATPSVM model analysis result (not that of direct SVM) is reported using Accuracy, Precision, Recall, False Acceptance Rate (FAR), False Rejection Rate (FRR), and Equal Error Rate (EER) [45].

- ✓ Accuracy= 100*(TA+TR)/(TA+TR+FA+FR) (65)
- ✓ Precision= 100*TA/(TA+FA) (66)
- ✓ Recall=100*TA/(TA+FR) (67)
- ✓ FAR=100*FA/(FA+TR) (68)
- ✓ FRR=100*FR/(FR+TA) (69)
- ✓ EER=FAR=FRR (70)

The same process is done for the different values of threshold *theta*. The optimal value for theta where best result of the model is obtained is also declared.

Exactly the same procedure is followed for all the ten Amharic words one by one. The average performance of the ATPSVM model for all the ten Amharic words experimented is also reported.

The FAR is the measure of the likelihood that this model will incorrectly accept an access attempt by an unauthorized user. The FRR is the measure of the probability that the model will fail to identify an individual who is properly enrolled.

6.5 Visual Inspection of the performance of the ATPSVM model

A visual inspection of the equal error rate (EER) is a commonly used evaluation tool in the speaker verification literature [2, 12, 14]. This is done using multi-line graphs for the FAR and FRR trends under similar threshold values. In this graph, the intersection point is the EER where type I error equals type II error. In the other areas, one of the trends increases while the other decreases.

The EER value indicates that the proportion of false acceptances is equal to the proportion of false rejections. The lower the equal error rate value, the higher the accuracy of the biometric system.

6.6 Factors affecting the performance of the ATPSVM model

The performance of speaker verification systems can be affected by many factors [2] mainly related to the speech signal such as the quality, modality, duration and population of the speech.

The quality of the speech can be affected by channel and microphone characteristics, noise level and type, and variability between enrolment and verification speech. Normalizations applied do reduce the mismatch effects.

The speech modality can be fixed-text, text-prompted, user-selected phrases or free text in which case text-prompted ones outperform. The ATPSVM model applies the last one.

Speech population size and composition also affect the performance of speaker verification systems. Speech duration and number of sessions of enrolment and verification speech also have effect on the performance of the new model. Better accuracy is obtained when more speech data is used for training and testing [2, 14]. The new model applies a total of 100 utterances of a speaker with ten each from the ten Amharic words experimented.

6.7 Analysis of the experimental result for the ATPSVM model

The experimentation is done using the prototype implemented for the ATPSVM model. Higher verification result is obtained by tuning the value of threshold *theta* in trial and error scheme. The 11 values of *theta* are taken to be 0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0 interchangeably and different performance results are obtained for the ATPSVM model.

The performance of the ATPSVM model is evaluated by using Linear, Gaussian Radial Basis Network, Polynomial and Multi-Layer Perceptron kernel functions and the results are reported. [45] is evaluated using RBF kernel function.

6.7.1 Using Linear Kernel function

The ATPSVM model is evaluated using linear kernel function but it does not converge. Thus this kernel cannot be applicable for the ATPSVM model.

6.7.2 Using Gaussian basis radial network kernel function (RBF)

The ATPSVM model is evaluated using this RBF kernel function and Table 6.3 shows the result obtained for the different threshold *theta* values. For [45], RBF is applied.

Table 6.3 Experimental result of the ATPSVM model using the RBF kernel

Theta	TEXT	TA	FA	FR	TR	TOTAL	ACCURACY	PRECISION	RECALL	FAR	FRR
0	1	900	900	0	0	1800	50	50	100	100	0
0	2	900	900	0	0	1800	50	50	100	100	0
0	3	900	900	0	0	1800	50	50	100	100	0
0	4	900	900	0	0	1800	50	50	100	100	0
0	5	900	900	0	0	1800	50	50	100	100	0
0	6	900	900	0	0	1800	50	50	100	100	0
0	7	900	900	0	0	1800	50	50	100	100	0
0	8	900	900	0	0	1800	50	50	100	100	0
0	9	900	900	0	0	1800	50	50	100	100	0
0	10	900	900	0	0	1800	50	50	100	100	0
0.1	1	879	466	21	434	1800	73	65	98	52	2
0.1	2	892	360	8	540	1800	80	71	99	40	1
0.1	3	900	290	0	610	1800	84	76	100	32	0
0.1	4	893	287	7	613	1800	84	76	99	32	1
0.1	5	897	215	3	685	1800	88	81	100	24	0

0.1	6	900	218	0	682	1800	88	81	100	24	0
0.1	7	897	209	3	691	1800	88	81	100	23	0
0.1	8	894	208	6	692	1800	88	81	99	23	1
0.1	9	896	207	4	693	1800	88	81	100	23	0
0.1	10	897	211	3	689	1800	88	81	100	23	0
0.2	1	864	435	36	465	1800	74	67	96	48	4
0.2	2	873	350	27	550	1800	79	71	97	39	3
0.2	3	895	266	5	634	1800	85	77	99	30	1
0.2	4	880	270	20	630	1800	84	77	98	30	2
0.2	5	885	204	15	696	1800	88	81	98	23	2
0.2	6	896	207	4	693	1800	88	81	100	23	0
0.2	7	892	198	8	702	1800	89	82	99	22	1
0.2	8	890	197	10	703	1800	89	82	99	22	1
0.2	9	890	203	10	697	1800	88	81	99	23	1
0.2	10	889	204	11	696	1800	88	81	99	23	1
0.3	1	820	384	80	516	1800	74	68	91	43	9
0.3	2	846	337	54	563	1800	78	72	94	37	6
0.3	3	854	245	46	655	1800	84	78	95	27	5
0.3	4	858	258	42	642	1800	83	77	95	29	5
0.3	5	871	193	29	707	1800	88	82	97	21	3
0.3	6	870	190	30	710	1800	88	82	97	21	3
0.3	7	875	182	25	718	1800	89	83	97	20	3
0.3	8	872	180	28	720	1800	88	83	97	20	3
0.3	9	875	191	25	709	1800	88	82	97	21	3
0.3	10	875	181	25	719	1800	89	83	97	20	3
0.4	1	758	319	142	581	1800	74	70	84	35	16
0.4	2	790	284	110	616	1800	78	74	88	32	12
0.4	3	787	219	113	681	1800	82	78	87	24	13
0.4	4	799	215	101	685	1800	82	79	89	24	11
0.4	5	817	166	83	734	1800	86	83	91	18	9
0.4	6	822	164	78	736	1800	87	83	91	18	9
0.4	7	826	163	74	737	1800	87	84	92	18	8
0.4	8	842	152	58	748	1800	88	85	94	17	6
0.4	9	835	161	65	739	1800	87	84	93	18	7
0.4	10	844	155	56	745	1800	88	84	94	17	6
0.5	01	670	211	230	689	1800	76	76	74	23	26
0.5	02	706	186	194	714	1800	79	79	78	21	22
0.5	03	772	104	128	796	1800	87	88	86	12	14
0.5	04	766	120	134	780	1800	86	86	85	13	15
0.5	**05**	**826**	**68**	**74**	**832**	**1800**	**92**	**92**	**92**	**8**	**8**
0.5	06	791	98	109	802	1800	89	89	88	11	12
0.5	07	798	91	102	809	1800	89	90	89	10	11
0.5	08	847	47	53	853	1800	94	95	94	5	6
0.5	09	827	61	73	839	1800	93	93	92	7	8

0.5	10	809	82	91	818	1800	90	91	90	9	10
0.6	1	581	142	319	758	1800	74	80	65	16	35
0.6	2	616	111	284	789	1800	78	85	68	12	32
0.6	3	681	113	219	787	1800	82	86	76	13	24
0.6	4	685	101	215	799	1800	82	87	76	11	24
0.6	5	734	83	166	817	1800	86	90	82	9	18
0.6	6	736	78	164	822	1800	87	90	82	9	18
0.6	7	736	73	164	827	1800	87	91	82	8	18
0.6	8	748	58	152	842	1800	88	93	83	6	17
0.6	9	739	65	161	835	1800	87	92	82	7	18
0.6	10	745	57	155	843	1800	88	93	83	6	17
0.7	1	516	80	384	820	1800	74	87	57	9	43
0.7	2	563	54	337	846	1800	78	91	63	6	37
0.7	3	655	46	245	854	1800	84	93	73	5	27
0.7	4	642	42	258	858	1800	83	94	71	5	29
0.7	5	707	29	193	871	1800	88	96	79	3	21
0.7	6	710	30	190	870	1800	88	96	79	3	21
0.7	7	718	25	182	875	1800	89	97	80	3	20
0.7	8	720	28	180	872	1800	88	96	80	3	20
0.7	9	709	25	191	875	1800	88	97	79	3	21
0.7	10	719	25	181	875	1800	89	97	80	3	20
0.8	1	465	36	435	864	1800	74	93	52	4	48
0.8	2	550	27	350	873	1800	79	95	61	3	39
0.8	3	636	5	264	895	1800	85	99	71	1	29
0.8	4	630	20	270	880	1800	84	97	70	2	30
0.8	5	696	15	204	885	1800	88	98	77	2	23
0.8	6	694	4	206	896	1800	88	99	77	0	23
0.8	7	702	7	198	893	1800	89	99	78	1	22
0.8	8	703	10	197	890	1800	89	99	78	1	22
0.8	9	697	10	203	890	1800	88	99	77	1	23
0.8	10	696	11	204	889	1800	88	98	77	1	23
0.9	1	434	21	466	879	1800	73	95	48	2	52
0.9	2	540	8	360	892	1800	80	99	60	1	40
0.9	3	610	0	290	900	1800	84	100	68	0	32
0.9	4	613	7	287	893	1800	84	99	68	1	32
0.9	5	684	3	216	897	1800	88	100	76	0	24
0.9	6	682	0	218	900	1800	88	100	76	0	24
0.9	7	691	3	209	897	1800	88	100	77	0	23
0.9	8	692	6	208	894	1800	88	99	77	1	23
0.9	9	693	4	207	896	1800	88	99	77	0	23
0.9	10	689	3	211	897	1800	88	100	77	0	23
1	1	362	7	538	893	1800	70	98	40	1	60
1	2	480	6	420	894	1800	76	99	53	1	47
1	3	535	0	365	900	1800	80	100	59	0	41
1	4	551	0	349	900	1800	81	100	61	0	39

1	5	654	0	246	900	1800	86	100	73	0	27
1	6	640	0	260	900	1800	86	100	71	0	29
1	7	642	0	258	900	1800	86	100	71	0	29
1	8	671	3	229	897	1800	87	100	75	0	25
1	9	674	4	226	896	1800	87	99	75	0	25
1	10	663	1	237	899	1800	87	100	74	0	26

Notice from Table 6.3 that better performance is obtained for the RBF kernel function at threshold *theta*=0.5 which is the shaded area in the table. This is where at least one EER (**8**) value is obtained. For this fixed threshold (theta=0.5), the average EER is 12.55% , average accuracy is 87.5%, average precision is 87.9%, and average recall is 86.8% for all the ten Amharic words experimented using the SVM RBF kernel for the ATPSVM model. Notice also the trend that FAR decreases while FRR increases down the RBF result table and Table 6.3 is easier to visualize it.

6.7.2.1 EER sketch

For EER sketch, the distinct Theta, FAR and FRR values are taken from Table 6.4 as a separate table labeled as Table 6.3.

Table 6.4: Distinct Theta, FAR and FRR values taken from Table 6.3 for EER sketch.

Theta	FAR	FRR
0	100	0
0.1	52	2
0.1	40	1
0.1	32	0
0.1	32	1
0.1	24	0
0.1	23	0
0.1	23	1
0.2	48	4
0.2	39	3
0.2	30	1
0.2	30	2
0.2	23	2
0.2	23	0
0.2	22	1
0.2	23	1
0.3	43	9
0.3	37	6
0.3	27	5
0.3	29	5
0.3	21	3
0.3	20	3
0.4	35	16
0.4	32	12
0.4	24	13
0.4	24	11
0.4	18	9
0.4	18	8
0.4	17	6
0.4	18	7
0.5	23	26
0.5	21	22
0.5	12	14
0.5	13	15
0.5	**8**	**8**
0.5	11	12
0.5	10	11
0.5	5	6
0.5	7	8
0.5	9	10
0.6	16	35
0.6	12	32
0.6	13	24
0.6	11	24
0.6	9	18
0.6	8	18
0.6	6	17
0.6	7	18
0.7	9	43
0.7	6	37
0.7	5	27
0.7	5	29
0.7	3	21
0.7	3	20
0.8	4	48

0.8	3	39
0.8	1	29
0.8	2	30
0.8	2	23
0.8	0	23
0.8	1	22

0.8	1	23
0.9	2	52
0.9	1	40
0.9	0	32
0.9	1	32
0.9	0	24

0.9	0	23
0.9	1	23
1	1	60
1	1	47
1	0	41
1	0	39

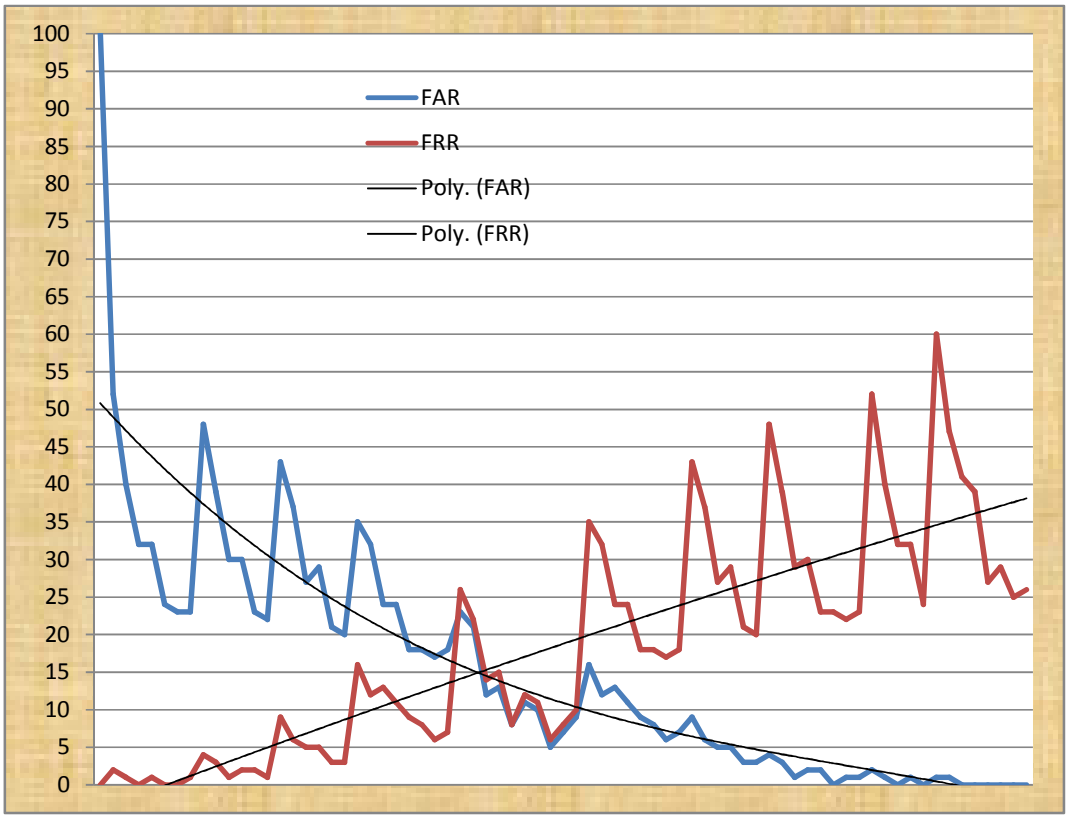

Figure 6.1: EER sketch for distinct Theta, FAR and FRR values of the RBF kernel.

Notice that the horizontal bar (x-axis) is the threshold value axis while the vertical (y-axis) is the Error Rate. The FAR goes down from the left-top corner down to the right bottom where as the FRR curve goes the opposite. Notice also that the exact EER point is at Error Rate 8 and not at 5. Between 5 and 7 the FRR is greater only by 1 with the *theta* values in that area being 0.5.

In addition, it is shown in Figure 6.1 that how the curves for accuracy and precision can be visualized in relation to the FAR and FRR curves.

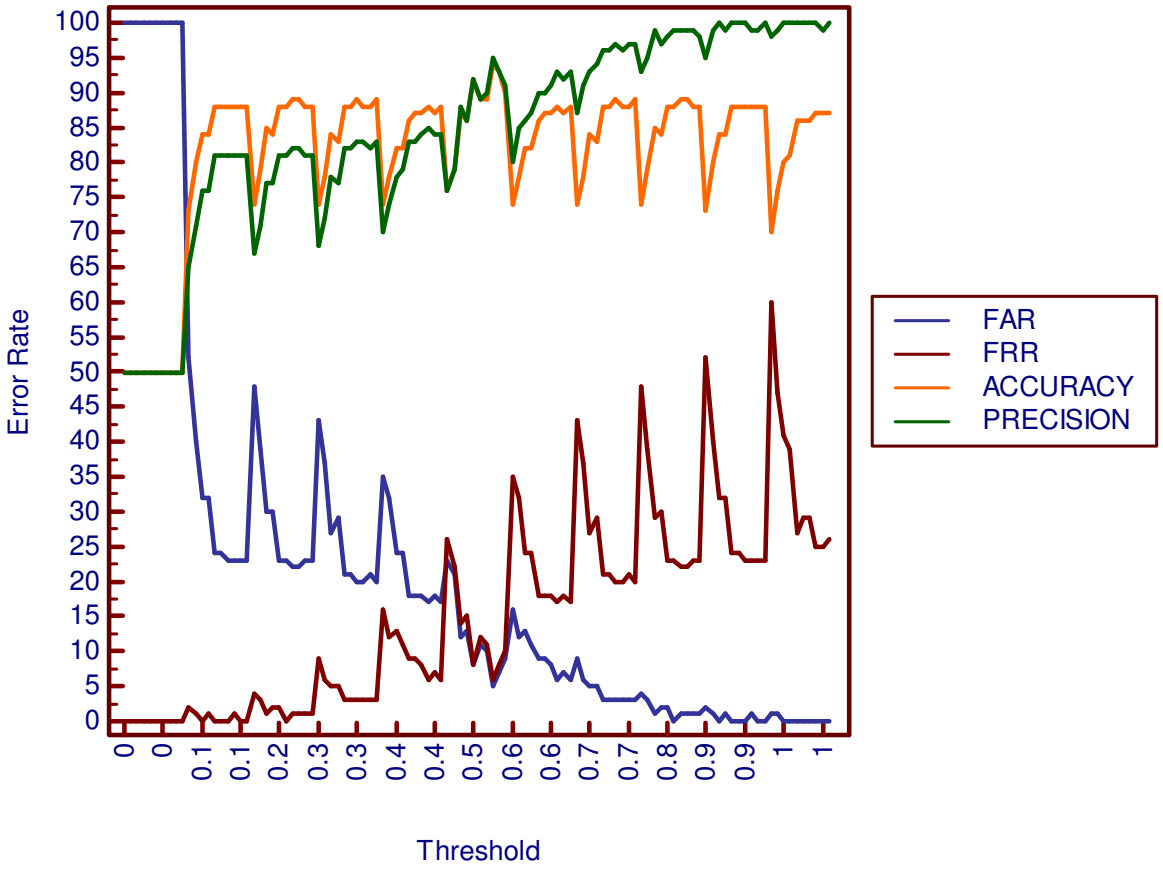

Figure 6.2: More inclusive visualization for the performance results using RBF.

Notice that precision increases as threshold theta value increases. Further by taking the RBF results of the new model only for threshold *theta*=0.5, the EER point can be clearly indicated as in Figure 6.3. Here FRR is always greater than FAR.

Text	FAR	FRR
1	5	6
2	7	8
3	8	8
4	9	10
5	10	11
6	11	12
7	12	14
8	13	15
9	21	22
10	23	26

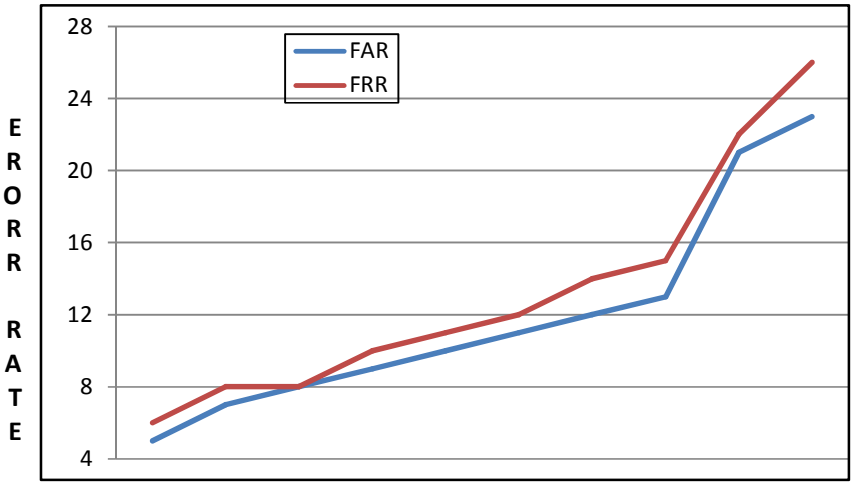

Figure 6.3: EER Sketch of the ATPSVM model for the RBF kernel at threshold *theta*=0.5.

6.7.3 Using Multi-Layer perceptron (MLP) kernel function

The ATPSVM model is evaluated using the SVM MLP kernel function and Table 6.5 shows the result obtained for the different *theta* values.

Table 6.5: Experimental result of the ATPSVM model using MLP kernel

Theta	TEXT	TA	FA	FR	TR	TOTAL	ACCURACY	PRECISION	RECALL	FAR	FRR
0	1	868	879	32	21	1800	49	50	96	98	4
0	2	888	889	12	11	1800	50	50	99	99	1
0	3	882	889	18	11	1800	50	50	98	99	2
0	4	870	875	30	25	1800	50	50	97	97	3
0	5	880	889	20	11	1800	50	50	98	99	2
0	6	878	878	22	22	1800	50	50	98	98	2
0	7	886	888	14	12	1800	50	50	98	99	2
0	8	885	888	15	12	1800	50	50	98	99	2
0	9	877	879	23	21	1800	50	50	97	98	3
0	10	887	891	13	9	1800	50	50	99	99	1
0.1	1	862	872	38	28	1800	49	50	96	97	4
0.1	2	886	887	14	13	1800	50	50	98	99	2
0.1	3	878	882	22	18	1800	50	50	98	98	2
0.1	4	864	866	36	34	1800	50	50	96	96	4
0.1	5	876	880	24	20	1800	50	50	97	98	3
0.1	6	875	877	25	23	1800	50	50	97	97	3

0.1	7	878	881	22	19	1800	50	50	98	98	2
0.1	8	879	882	21	18	1800	50	50	98	98	2
0.1	9	873	875	27	25	1800	50	50	97	97	3
0.1	10	884	886	16	14	1800	50	50	98	98	2
0.2	1	855	849	45	51	1800	50	50	95	94	5
0.2	2	885	828	15	72	1800	53	52	98	92	2
0.2	3	872	876	28	24	1800	50	50	97	97	3
0.2	4	861	849	39	51	1800	51	50	96	94	4
0.2	5	868	872	32	28	1800	50	50	96	97	4
0.2	6	874	837	26	63	1800	52	51	97	93	3
0.2	7	876	875	24	25	1800	50	50	97	97	3
0.2	8	877	873	23	27	1800	50	50	97	97	3
0.2	9	869	866	31	34	1800	50	50	97	96	3
0.2	10	881	863	19	37	1800	51	51	98	96	2
0.3	1	838	675	62	225	1800	59	55	93	75	7
0.3	2	869	574	31	326	1800	66	60	97	64	3
0.3	3	836	819	64	81	1800	51	51	93	91	7
0.3	4	839	691	61	209	1800	58	55	93	77	7
0.3	5	853	777	47	123	1800	54	52	95	86	5
0.3	6	869	650	31	250	1800	62	57	97	72	3
0.3	7	868	787	32	113	1800	55	52	96	87	4
0.3	8	859	771	41	129	1800	55	53	95	86	5
0.3	9	859	647	41	253	1800	62	57	95	72	5
0.3	10	871	712	29	188	1800	59	55	97	79	3
0.4	1	787	375	113	525	1800	73	68	87	42	13
0.4	2	833	279	67	621	1800	81	75	93	31	7
0.4	3	732	615	168	285	1800	57	54	81	68	19
0.4	4	789	386	111	514	1800	72	67	88	43	12
0.4	5	793	500	107	400	1800	66	61	88	56	12
0.4	6	813	388	87	512	1800	74	68	90	43	10
0.4	7	816	458	84	442	1800	70	64	91	51	9
0.4	8	799	495	101	405	1800	67	62	89	55	11
0.4	9	832	287	68	613	1800	80	74	92	32	8
0.4	10	822	408	78	492	1800	73	67	91	45	9
0.48	**1**	**700**	**202**	**200**	**698**	**1800**	**78**	**78**	**78**	**22**	**22**
0.48	2	776	131	124	769	1800	86	86	86	15	14
0.48	3	572	378	328	522	1800	61	60	64	42	36
0.48	4	686	197	214	703	1800	77	78	76	22	24
0.48	5	669	251	231	649	1800	73	73	74	28	26
0.48	6	713	210	187	690	1800	78	77	79	23	21
0.48	7	691	243	209	657	1800	75	74	77	27	23
0.48	8	677	264	223	636	1800	73	72	75	29	25
0.48	**9**	**779**	**119**	**121**	**781**	**1800**	**87**	**87**	**87**	**13**	**13**

0.48	10	720	211	180	689	1800	78	77	80	23	20
0.5	1	660	156	240	744	1800	78	81	73	17	27
0.5	2	738	107	162	793	1800	85	87	82	12	18
0.5	3	497	311	403	589	1800	60	62	55	35	45
0.5	4	637	160	263	740	1800	77	80	71	18	29
0.5	5	620	196	280	704	1800	74	76	69	22	31
0.5	6	657	167	243	733	1800	77	80	73	19	27
0.5	7	640	186	260	714	1800	75	77	71	21	29
0.5	8	606	198	294	702	1800	73	75	67	22	33
0.5	9	759	98	141	802	1800	87	89	84	11	16
0.5	10	669	167	231	733	1800	78	80	74	19	26
0.6	1	471	63	429	837	1800	73	88	52	7	48
0.6	2	592	48	308	852	1800	80	93	66	5	34
0.6	3	257	135	643	765	1800	57	66	29	15	71
0.6	4	463	79	437	821	1800	71	85	51	9	49
0.6	5	381	70	519	830	1800	67	84	42	8	58
0.6	6	475	61	425	839	1800	73	89	53	7	47
0.6	7	426	59	474	841	1800	70	88	47	7	53
0.6	8	378	79	522	821	1800	67	83	42	9	58
0.6	9	594	46	306	854	1800	80	93	66	5	34
0.6	10	473	62	427	838	1800	73	88	53	7	47
0.7	1	168	18	732	882	1800	58	90	19	2	81
0.7	2	313	14	587	886	1800	67	96	35	2	65
0.7	3	59	44	841	856	1800	51	57	7	5	93
0.7	4	174	19	726	881	1800	59	90	19	2	81
0.7	5	87	19	813	881	1800	54	82	10	2	90
0.7	6	205	7	695	893	1800	61	97	23	1	77
0.7	7	105	9	795	891	1800	55	92	12	1	88
0.7	8	93	13	807	887	1800	54	88	10	1	90
0.7	9	220	16	680	884	1800	61	93	24	2	76
0.7	10	164	11	736	889	1800	59	94	18	1	82
0.8	1	12	2	888	898	1800	51	86	1	0	99
0.8	2	57	1	843	899	1800	53	98	6	0	94
0.8	3	4	7	896	893	1800	50	36	0	1	100
0.8	4	15	3	885	897	1800	51	83	2	0	98
0.8	5	7	4	893	896	1800	50	64	1	0	99
0.8	6	28	0	872	900	1800	52	100	3	0	97
0.8	7	8	1	892	899	1800	50	89	1	0	99
0.8	8	6	1	894	899	1800	50	86	1	0	99
0.8	9	12	3	888	897	1800	51	80	1	0	99
0.8	10	25	0	875	900	1800	51	100	3	0	97
0.9	1	0	1	900	899	1800	50	0	0	0	100
0.9	2	0	0	900	900	1800	50	0	0	0	100

0.9	3	0	0	900	900	1800	50	0	0	0	100
0.9	4	0	0	900	900	1800	50	0	0	0	100
0.9	5	0	0	900	900	1800	50	0	0	0	100
0.9	6	0	0	900	900	1800	50	0	0	0	100
0.9	7	0	0	900	900	1800	50	0	0	0	100
0.9	8	0	0	900	900	1800	50	0	0	0	100
0.9	9	0	0	900	900	1800	50	0	0	0	100
0.9	10	0	0	900	900	1800	50	0	0	0	100
1	1	0	0	900	900	1800	50	0	0	0	100
1	2	0	0	900	900	1800	50	0	0	0	100
1	3	0	0	900	900	1800	50	0	0	0	100
1	4	0	0	900	900	1800	50	0	0	0	100
1	5	0	0	900	900	1800	50	0	0	0	100
1	6	0	0	900	900	1800	50	0	0	0	100
1	7	0	0	900	900	1800	50	0	0	0	100
1	8	0	0	900	900	1800	50	0	0	0	100
1	9	0	0	900	900	1800	50	0	0	0	100
1	10	0	0	900	900	1800	50	0	0	0	100

Notice that better performance of the ATPSVM model is obtained for the MLP kernel function at *theta=0.48* where two EER points (**13** & **22**) are indicated in bold. The threshold value *theta=0.48* is obtained after continuous trial and error efforts, otherwise no EER value for *theta=0.5* or *theta=0.6*. In addition, 23.4% EER, 76.6% accuracy, 76.2% precision and 77.6% recall average values are obtained for the ATPSVM model for all the Amharic words experimented using the SVM MLP kernel function. And this implies that the ATPSVM model has performed much better with the SVM RBF kernel than the SVM MLP kernel function.

6.7.3.1 EER Sketch

For EER sketch, the distinct Theta, FAR and FRR values are taken from Table 6.5 as a separate table labeled as Table 6.6.

Table 6.6: Distinct Theta, FAR and FRR values taken from Table 6.5 for EER sketch.

Theta	FAR	FRR
0	98	4
0	99	1
0	99	2
0	97	3
0	98	2
0	98	3
0.1	97	4
0.1	99	2
0.1	98	2
0.1	96	4
0.1	98	3
0.1	97	3
0.2	94	5
0.2	92	2
0.2	97	3
0.2	94	4
0.2	97	4
0.2	93	3
0.2	96	3
0.2	96	2
0.3	75	7
0.3	64	3
0.3	91	7
0.3	77	7
0.3	86	5
0.3	72	3
0.3	87	4
0.3	72	5
0.3	79	3
0.4	42	13
0.4	31	7
0.4	68	19
0.4	43	12
0.4	56	12
0.4	43	10
0.4	51	9
0.4	55	11
0.4	32	8
0.4	45	9
0.48	**22**	**22**
0.48	15	14
0.48	42	36
0.48	22	24
0.48	28	26
0.48	23	21
0.48	27	23
0.48	29	25
0.48	**13**	**13**
0.48	23	20
0.5	17	27
0.5	12	18
0.5	35	45
0.5	18	29
0.5	22	31
0.5	19	27
0.5	21	29
0.5	22	33
0.5	11	16
0.5	19	26
0.6	7	48
0.6	5	34
0.6	15	71
0.6	9	49
0.6	8	58
0.6	7	47
0.6	7	53
0.6	9	58
0.7	2	81
0.7	2	65
0.7	5	93
0.7	2	90
0.7	1	77
0.7	1	88
0.7	1	90
0.7	2	76
0.7	1	82

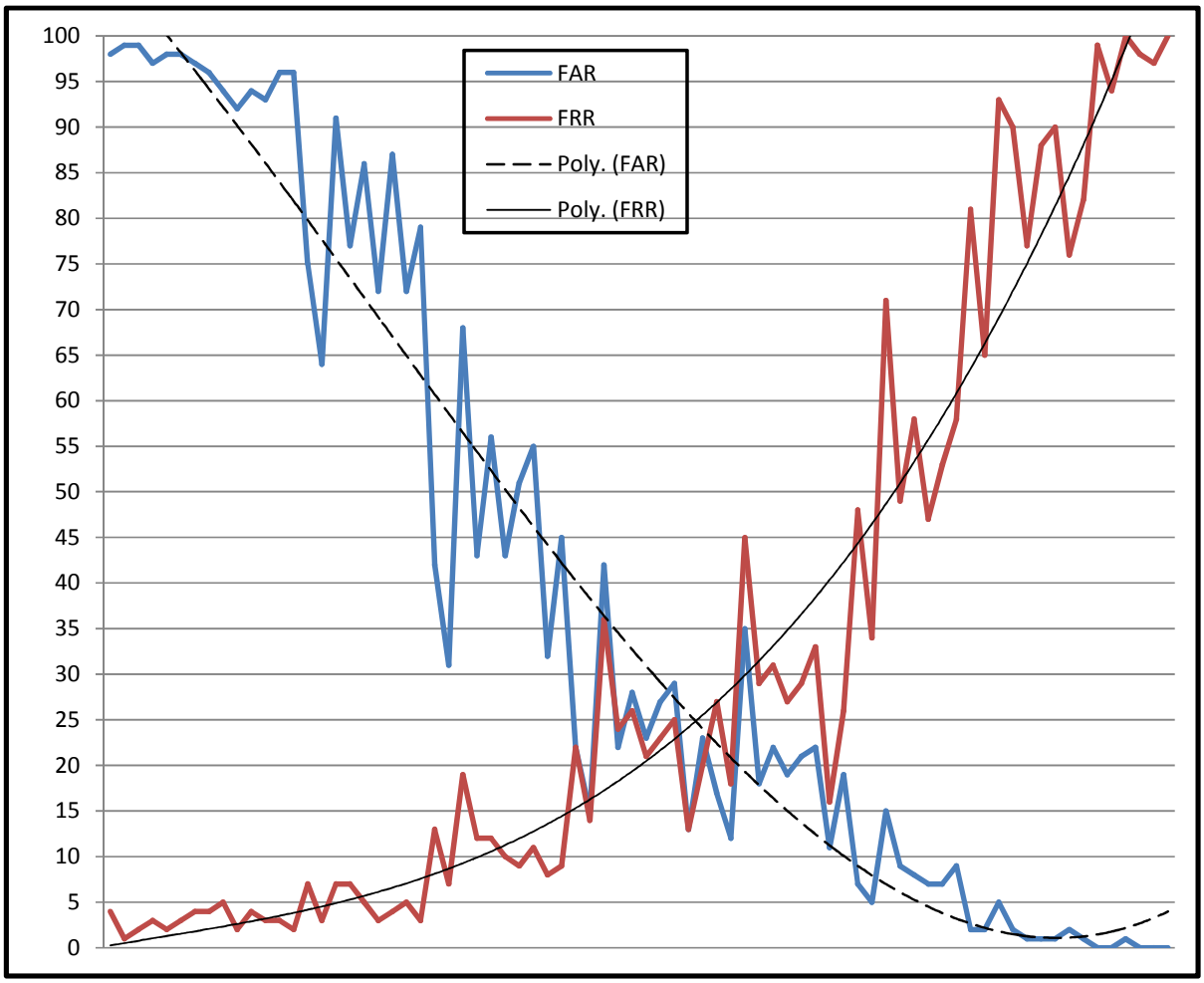

Figure 6.4 EER sketch for the ATPSVM model using MLP kernel function at all *theta* values.

Notice that the trend lines indicate the nature of the FAR and FRR curves as a function of *theta*. Finally, the FAR and FRR values at *theta=0.48* are taken and the EER sketch is shown in Figure 6.5.

Text	FAR	FRR
1	13	13
2	15	14
3	22	22
4	22	24
5	23	21
6	23	20
7	27	23
8	28	26
9	29	25
10	42	36

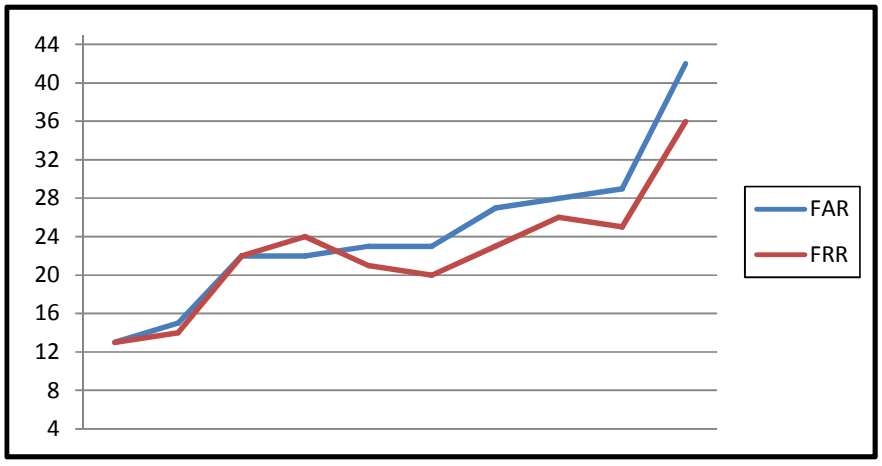

Figure 6.5: EER sketch for the ATPSVM model using SVM MLP kernel function for *theta*=0.48.

6.7.4 Using SVM Polynomial of power 3 kernel function

The ATPSVM model is evaluated using this Polynomial of power 3 kernel function and Table 6.7 shows the result obtained for the different *theta* values.

Table 6.7: Experimental result of the ATPSVM model using Polynomial kernel

Theta	TEXT	TA	FA	FR	TR	TOTAL	ACCURACY	PRECISION	RECALL	FAR	FRR
0	1	900	778	0	122	1800	57	54	100	86	0
0	2	900	717	0	183	1800	60	56	100	80	0
0	3	900	598	0	302	1800	67	60	100	66	0
0	4	900	612	0	288	1800	66	60	100	68	0
0	5	900	488	0	412	1800	73	65	100	54	0
0	6	900	556	0	344	1800	69	62	100	62	0
0	7	900	536	0	364	1800	70	63	100	60	0
0	8	900	431	0	469	1800	76	68	100	48	0
0	9	900	474	0	426	1800	74	66	100	53	0
0	10	900	490	0	410	1800	73	65	100	54	0
0.1	1	900	354	0	546	1800	80	72	100	39	0
0.1	2	900	317	0	583	1800	82	74	100	35	0
0.1	3	900	269	0	631	1800	85	77	100	30	0
0.1	4	900	256	0	644	1800	86	78	100	28	0
0.1	5	900	172	0	728	1800	90	84	100	19	0
0.1	6	900	224	0	676	1800	88	80	100	25	0
0.1	7	900	189	0	711	1800	90	83	100	21	0
0.1	8	900	117	0	783	1800	94	88	100	13	0
0.1	9	900	132	0	768	1800	93	87	100	15	0
0.1	10	900	156	0	744	1800	91	85	100	17	0
0.2	1	900	113	0	787	1800	94	89	100	13	0

0.2	2	900	86	0	814	1800	95	91	100	10	0
0.2	3	900	101	0	799	1800	94	90	100	11	0
0.2	4	900	62	0	838	1800	97	94	100	7	0
0.2	5	900	40	0	860	1800	98	96	100	4	0
0.2	6	900	58	0	842	1800	97	94	100	6	0
0.2	7	900	54	0	846	1800	97	94	100	6	0
0.2	8	900	15	0	885	1800	99	98	100	2	0
0.2	9	900	35	0	865	1800	98	96	100	4	0
0.2	10	900	44	0	856	1800	98	95	100	5	0
0.3	1	899	52	1	848	1800	97	95	100	6	0
0.3	2	898	30	2	870	1800	98	97	100	3	0
0.3	3	900	27	0	873	1800	99	97	100	3	0
0.3	4	900	20	0	880	1800	99	98	100	2	0
0.3	5	900	9	0	891	1800	100	99	100	1	0
0.3	6	900	17	0	883	1800	99	98	100	2	0
0.3	7	900	14	0	886	1800	99	98	100	2	0
0.3	8	900	3	0	897	1800	100	100	100	0	0
0.3	9	899	12	1	888	1800	99	99	100	1	0
0.3	10	900	16	0	884	1800	99	98	100	2	0
0.4	1	895	17	5	883	1800	99	98	99	2	1
0.4	2	897	11	3	889	1800	99	99	100	1	0
0.4	3	900	6	0	894	1800	100	99	100	1	0
0.4	4	900	6	0	894	1800	100	99	100	1	0
0.4	5	900	3	0	897	1800	100	100	100	0	0
0.4	6	900	4	0	896	1800	100	100	100	0	0
0.4	7	900	4	0	896	1800	100	100	100	0	0
0.4	8	900	1	0	899	1800	100	100	100	0	0
0.4	9	898	8	2	892	1800	99	99	100	1	0
0.4	10	900	5	0	895	1800	100	99	100	1	0
0.5	**1**	**890**	**9**	**10**	**891**	**1800**	**99**	**99**	**99**	**1**	**1**
0.5	**2**	**892**	**5**	**8**	**895**	**1800**	**99**	**99**	**99**	**1**	**1**
0.5	**3**	**898**	**1**	**2**	**899**	**1800**	**100**	**100**	**100**	**0**	**0**
0.5	**4**	**898**	**2**	**2**	**898**	**1800**	**100**	**100**	**100**	**0**	**0**
0.5	**5**	**900**	**0**	**0**	**900**	**1800**	**100**	**100**	**100**	**0**	**0**
0.5	**6**	**899**	**0**	**1**	**900**	**1800**	**100**	**100**	**100**	**0**	**0**
0.5	**7**	**898**	**1**	**2**	**899**	**1800**	**100**	**100**	**100**	**0**	**0**
0.5	**8**	**900**	**0**	**0**	**900**	**1800**	**100**	**100**	**100**	**0**	**0**
0.5	9	894	4	6	896	1800	99	100	99	0	1
0.5	**10**	**899**	**1**	**1**	**899**	**1800**	**100**	**100**	**100**	**0**	**0**
0.6	1	883	5	17	895	1800	99	99	98	1	2
0.6	2	889	3	11	897	1800	99	100	99	0	1
0.6	3	894	0	6	900	1800	100	100	99	0	1
0.6	4	894	0	6	900	1800	100	100	99	0	1
0.6	5	897	0	3	900	1800	100	100	100	0	0

0.6	6	896	0	4	900	1800	100	100	100	0	0
0.6	7	896	0	4	900	1800	100	100	100	0	0
0.6	8	899	0	1	900	1800	100	100	100	0	0
0.6	9	892	2	8	898	1800	99	100	99	0	1
0.6	10	895	0	5	900	1800	100	100	99	0	1
0.7	1	848	1	52	899	1800	97	100	94	0	6
0.7	2	870	2	30	898	1800	98	100	97	0	3
0.7	3	873	0	27	900	1800	99	100	97	0	3
0.7	4	880	0	20	900	1800	99	100	98	0	2
0.7	5	891	0	9	900	1800	100	100	99	0	1
0.7	6	883	0	17	900	1800	99	100	98	0	2
0.7	7	886	0	14	900	1800	99	100	98	0	2
0.7	8	897	0	3	900	1800	100	100	100	0	0
0.7	9	888	1	12	899	1800	99	100	99	0	1
0.7	10	884	0	16	900	1800	99	100	98	0	2
0.8	1	787	0	113	900	1800	94	100	87	0	13
0.8	2	814	0	86	900	1800	95	100	90	0	10
0.8	3	799	0	101	900	1800	94	100	89	0	11
0.8	4	838	0	62	900	1800	97	100	93	0	7
0.8	5	859	0	41	900	1800	98	100	95	0	5
0.8	6	841	0	59	900	1800	97	100	93	0	7
0.8	7	846	0	54	900	1800	97	100	94	0	6
0.8	8	885	0	15	900	1800	99	100	98	0	2
0.8	9	865	0	35	900	1800	98	100	96	0	4
0.8	10	856	0	44	900	1800	98	100	95	0	5
0.9	1	546	0	354	900	1800	80	100	61	0	39
0.9	2	583	0	317	900	1800	82	100	65	0	35
0.9	3	631	0	269	900	1800	85	100	70	0	30
0.9	4	641	0	259	900	1800	86	100	71	0	29
0.9	5	727	0	173	900	1800	90	100	81	0	19
0.9	6	675	0	225	900	1800	88	100	75	0	25
0.9	7	710	0	190	900	1800	89	100	79	0	21
0.9	8	781	0	119	900	1800	93	100	87	0	13
0.9	9	768	0	132	900	1800	93	100	85	0	15
0.9	10	743	0	157	900	1800	91	100	83	0	17
1	1	0	0	900	900	1800	50	0	0	0	100
1	2	0	0	900	900	1800	50	0	0	0	100
1	3	0	0	900	900	1800	50	0	0	0	100
1	4	0	0	900	900	1800	50	0	0	0	100
1	5	0	0	900	900	1800	50	0	0	0	100
1	6	0	0	900	900	1800	50	0	0	0	100
1	7	0	0	900	900	1800	50	0	0	0	100
1	8	0	0	900	900	1800	50	0	0	0	100
1	9	0	0	900	900	1800	50	0	0	0	100
1	10	0	0	900	900	1800	50	0	0	0	100

Best performance of the ATPSVM model is obtained at *theta=0.5* using the SVM polynomial kernel function. This accuracy increment is attributed to the algorithmic definition of the SVM polynomial kernel function that maps the training data to the kernel space. The average values for all the Amharic words experimented using the SVM polynomial kernel function at theta=0.5 are 99.7% accuracy, 99.8% Precision, 99.7% Recall and 0.25% EER. This is the best experimental result obtained for the ATPSVM model while evaluating its performance using the different kernel functions so far.

6.7.4.1 EER Sketch

For EER sketch, the distinct Theta, FAR and FRR values are taken from Table 6.7 as a separate table labeled as Table 6.8.

Table 6.8: Distinct Theta, FAR and FRR values taken from Table 6.7 for EER sketch.

Theta	FAR	FRR
0	86	0
0	80	0
0	66	0
0	68	0
0	54	0
0	62	0
0	60	0
0	48	0
0	53	0
0.1	39	0
0.1	35	0
0.1	30	0
0.1	28	0
0.1	19	0
0.1	25	0
0.1	21	0
0.1	13	0
0.1	15	0
0.1	17	0
0.2	13	0
0.2	10	0
0.2	11	0
0.2	7	0
0.2	4	0
0.2	6	0
0.2	2	0
0.2	5	0
0.3	6	0
0.3	3	0
0.3	2	0
0.3	1	0
0.3	**0**	**0**
0.4	2	1
0.4	1	0
0.4	**0**	**0**
0.5	**1**	**1**
0.5	**0**	**0**
0.5	0	1
0.6	1	2
0.6	0	1
0.6	**0**	**0**
0.7	0	6
0.7	0	3
0.7	0	2
0.7	0	1
0.7	**0**	**0**
0.8	0	13
0.8	0	10
0.8	0	11
0.8	0	7
0.8	0	5
0.8	0	6
0.8	0	2
0.8	0	4
0.9	0	39
0.9	0	35
0.9	0	30
0.9	0	29
0.9	0	19
0.9	0	25
0.9	0	21
0.9	0	13
0.9	0	15
0.9	0	17
1	0	100

Notice EER values are obtained at theta values of 0.3, 0.4, 0.5, 0.6 & 0.7. All the performance metrics are depicted in Figure 6.6 and the trend lines are depicted in Figure 6.7.

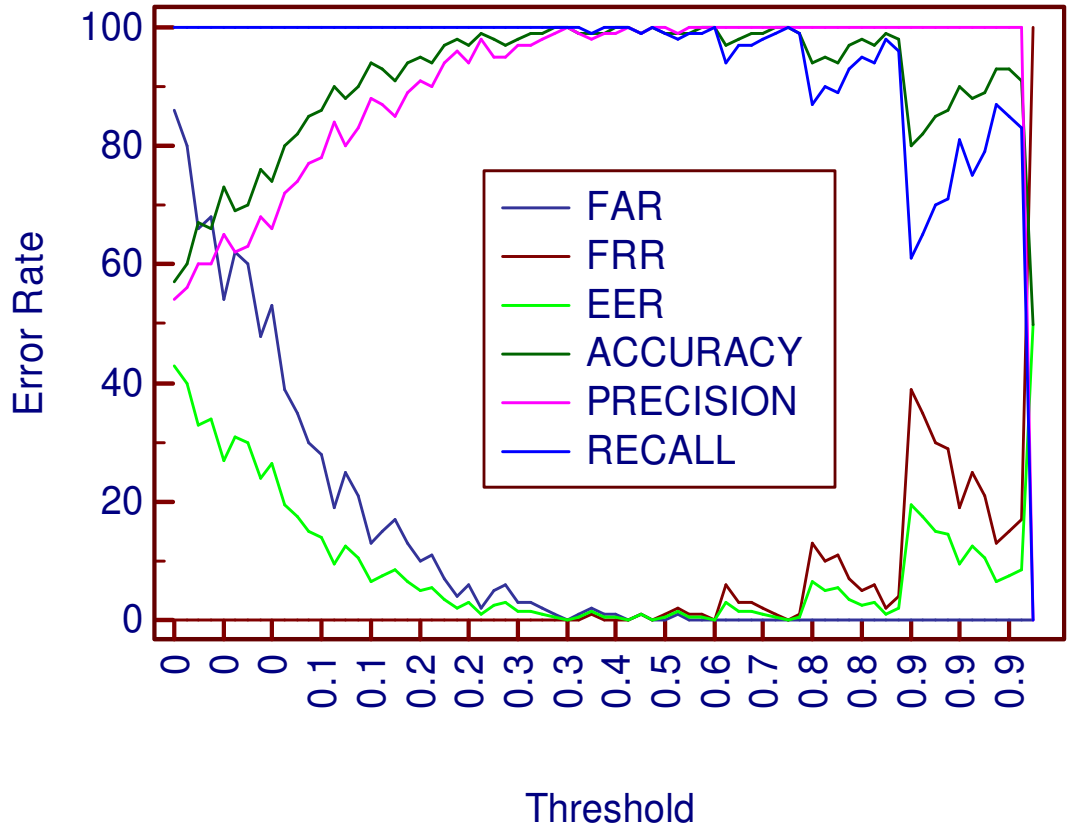

Figure 6.6 The performance metrics for the ATPSVM model using the SVM Polynomial Kernel.

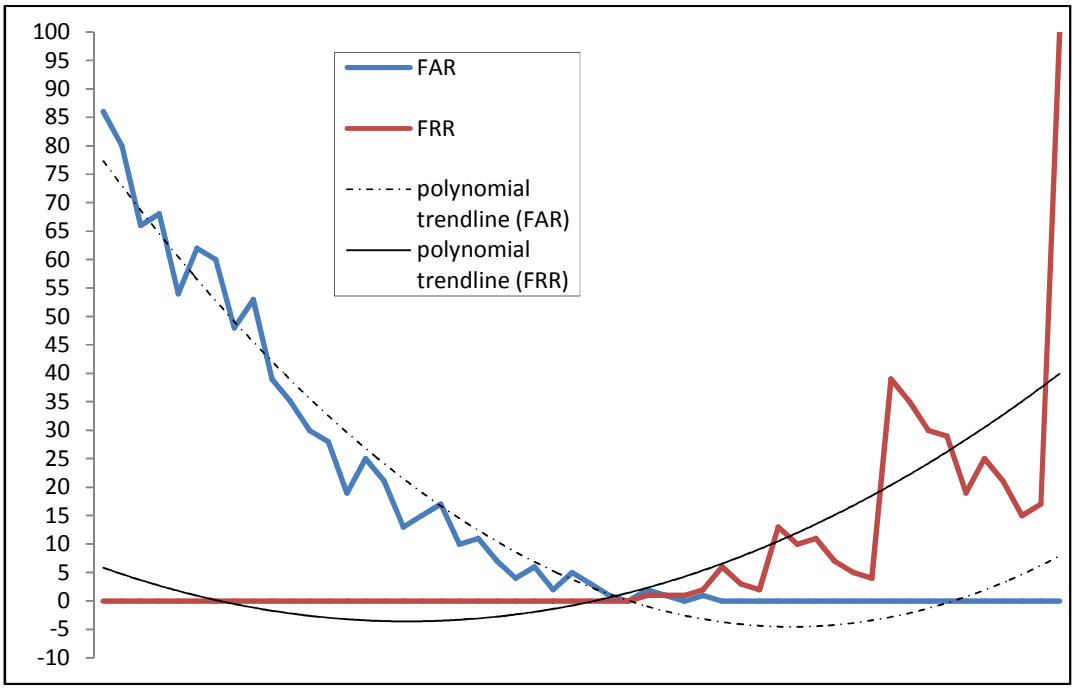

Figure 6.7 EER sketch trend lines for the ATPSVM model using the SVM Polynomial Kernel.

Further, the FAR and FRR values at *theta=0.5* are taken and the respective EER sketch is shown in Figure 6.8.

Text	FAR	FRR
1	1	1
2	1	1
3	0	0
4	0	0
5	0	0
6	0	0
7	0	0
8	0	0
9	0	1
10	0	0

ERROR RATE

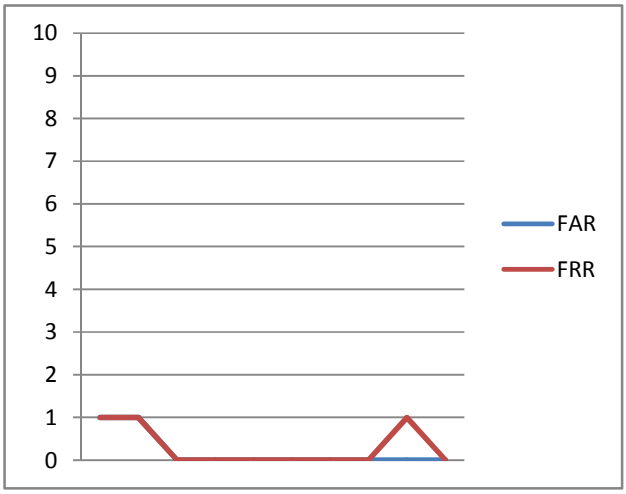

Figure 6.8: The EER sketch for the SVM Polynomial order 3 Kernel at theta=0.5.

Notice that the Error rate of the new model is Zero or closer to it for the polynomial kernel.

6.7.5 Summary of the Experimental result for the ATPSVM model

Table 6.9: Source table for average performance of each SVM Kernel

Text	Accuracy	Precision	Recall	FAR	FRR	Avg. EER	Kernel
1	78	78	78	22	22		
2	86	86	86	15	14		
3	61	60	64	42	36		
4	77	78	76	22	24		
5	73	73	74	28	26		
6	78	77	79	23	21		
7	75	74	77	27	23		
8	73	72	75	29	25		
9	87	87	87	13	13		
10	78	77	80	23	20		
Average	**76.6**	**76.2**	**77.6**	**24.4**	**22.4**	**23.4**	**MLP (theta=0.48)**
1	76	76	74	23	26		
2	79	79	78	21	22		
3	87	88	86	12	14		
4	86	86	85	13	15		
5	92	92	92	8	8		
6	89	89	88	11	12		
7	89	90	89	10	11		
8	94	95	94	5	6		
9	93	93	92	7	8		
10	90	91	90	9	10		
Average	**87.5**	**87.9**	**86.8**	**11.9**	**13.2**	**12.55**	**RBF(theta=0.5)**
1	99	99	99	1	1		
2	99	99	99	1	1		
3	100	100	100	0	0		
4	100	100	100	0	0		
5	100	100	100	0	0		
6	100	100	100	0	0		
7	100	100	100	0	0		
8	100	100	100	0	0		
9	99	100	99	0	1		
10	100	100	100	0	0		
Average	**99.7**	**99.8**	**99.7**	**0.2**	**0.3**	**0.25**	**Polynomial (theta=0.5)**

The average summary result obtained from Table 6.9 is reported in Table 6.10.

Table 6.10 Average performance summary of the ATPSVM model for each kernel

Kernel	EER	Accuracy	Precision	Recall	theta
MLP	23.4%	76.6%	76.2%	77.6%	0.48
RBF	12.55%	87.5%	87.9%	86.8%	0.5
Polynomial	0.25%	99.7%	99.8%	99.7%	0.5

Thus, Table 6.10 shows that the best result of the ATPSVM model is obtained when the Polynomial order 3 kernel is used.

So it is found important to investigate the performance of the ATPSVM model for each Amharic word separately for polynomial kernel and it is presented in Section 6.8.

6.8 SVM Polynomial Kernel result Analysis by Amharic Word

Further analysis is done on the experimental result of the ATPSVM model to determine the discriminative capability of each Amharic word used in the experiment. The respective EER sketch is depicted for each word.

6.8.1 SVM Polynomial Kernel result Analysis for the Amharic Word coded '01'

Table 6.11 Experimental result for Amharic word code '01'

Theta	TEXT	TA	FA	FR	TR	TOTAL	ACCURACY	PRECISION	RECALL	FAR	FRR
0	1	900	778	0	122	1800	57	54	100	86	0
0.1	1	900	354	0	546	1800	80	72	100	39	0
0.2	1	900	113	0	787	1800	94	89	100	13	0
0.4	1	895	17	5	883	1800	99	98	99	2	1
0.5	1	890	9	10	891	1800	99	99	99	1	1
0.6	1	883	5	17	895	1800	99	99	98	1	2
0.7	1	848	1	52	899	1800	97	100	94	0	6
0.8	1	787	0	113	900	1800	94	100	87	0	13
0.9	1	546	0	354	900	1800	80	100	61	0	39
1	1	0	0	900	900	1800	50	0	0	0	100

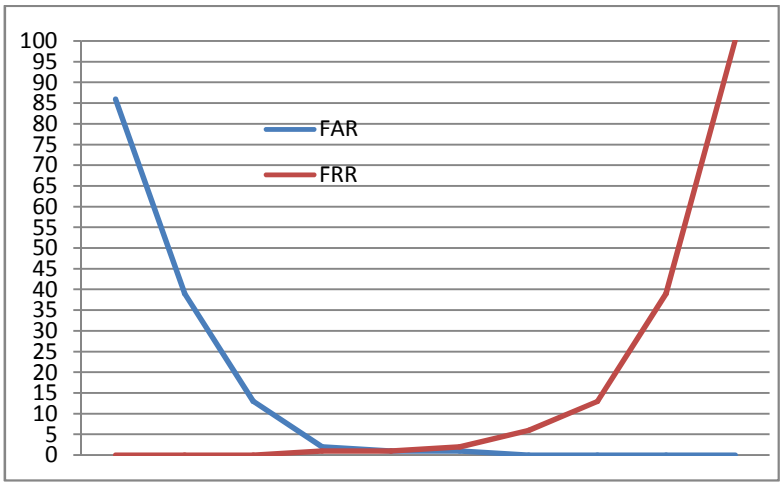

Figure 6.9 EER sketch for the experimental result of the Amharic word code '01'

- Maximum accuracy obtained for Amharic word coded '01' is 99% with 99% precision, 99% recall and 1% EER at 0.5 theta value.

6.8.2 SVM Polynomial Kernel result Analysis for the Amharic Word coded '02'

Table 6.12 Experimental result for Amharic word code '02'

Theta	TEXT	TA	FA	FR	TR	TOTAL	ACCURACY	PRECISION	RECALL	FAR	FRR
0	2	900	717	0	183	1800	60	56	100	80	0
0.1	2	900	317	0	583	1800	82	74	100	35	0
0.2	2	900	86	0	814	1800	95	91	100	10	0
0.4	2	897	11	3	889	1800	99	99	100	1	0
0.5	2	892	5	8	895	1800	99	99	99	1	1
0.6	2	889	3	11	897	1800	99	100	99	0	1
0.7	2	870	2	30	898	1800	98	100	97	0	3
0.8	2	814	0	86	900	1800	95	100	90	0	10
0.9	2	583	0	317	900	1800	82	100	65	0	35
1	2	0	0	900	900	1800	50	0	0	0	100

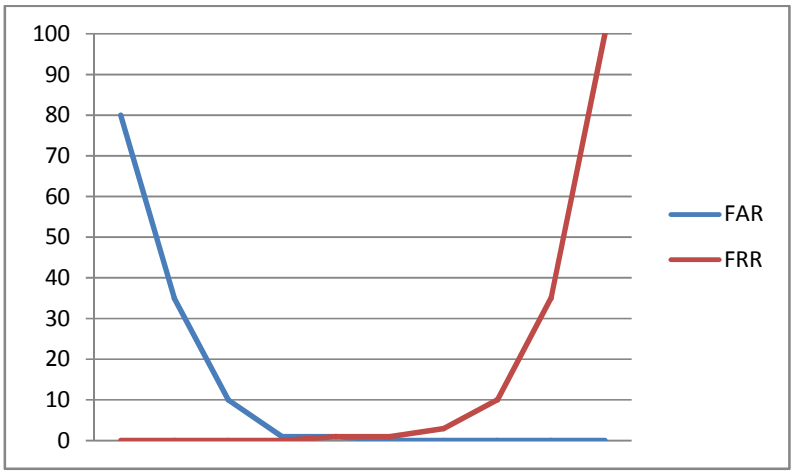

Figure 6.10 EER sketch for the experimental result of the Amharic word code '02'

- Maximum accuracy obtained for Amharic word coded '02' is 99% with 100% precision, 99% recall and 0.5% averaged EER at 0.6 theta value.

6.8.3 SVM Polynomial Kernel result Analysis for the Amharic Word coded '03'

Table 6.13 Experimental result for Amharic word code '03'

Theta	TEXT	TA	FA	FR	TR	TOTAL	ACCURACY	PRECISION	RECALL	FAR	FRR
0	3	900	598	0	302	1800	67	60	100	66	0
0.1	3	900	269	0	631	1800	85	77	100	30	0
0.2	3	900	101	0	799	1800	94	90	100	11	0
0.4	3	900	6	0	894	1800	100	99	100	1	0
0.5	3	898	1	2	899	1800	100	100	100	0	0
0.6	3	894	0	6	900	1800	100	100	99	0	1
0.7	3	873	0	27	900	1800	99	100	97	0	3
0.8	3	799	0	101	900	1800	94	100	89	0	11
0.9	3	631	0	269	900	1800	85	100	70	0	30
1	3	0	0	900	900	1800	50	0	0	0	100

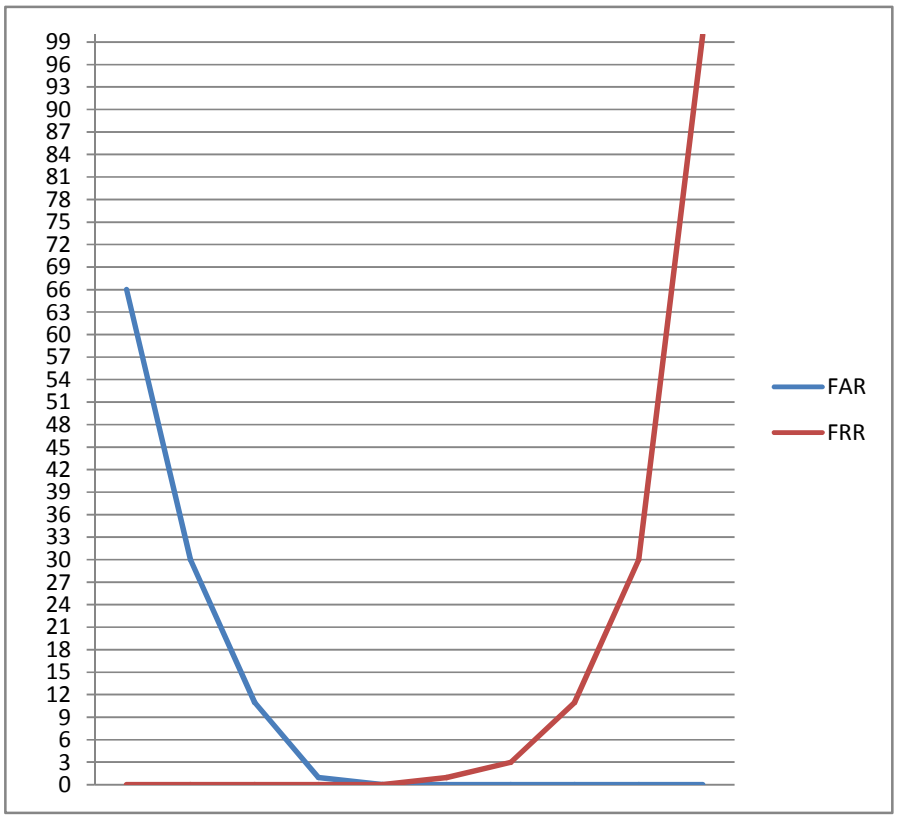

Figure 6.11 EER sketch for the experimental result of the Amharic word code '03'

- Maximum accuracy obtained for Amharic word coded '03' is 100% with 100% precision, 100% recall and 0.00% EER at 0.5 theta value.

6.8.4 SVM Polynomial Kernel result Analysis for the Amharic Word coded '04'

Table 6.14 Experimental result for Amharic word code '04'

Theta	TEXT	TA	FA	FR	TR	TOTAL	ACCURACY	PRECISION	RECALL	FAR	FRR
0	4	900	612	0	288	1800	66	60	100	68	0
0.1	4	900	256	0	644	1800	86	78	100	28	0
0.2	4	900	62	0	838	1800	97	94	100	7	0
0.4	4	900	6	0	894	1800	100	99	100	1	0
0.5	4	898	2	2	898	1800	100	100	100	0	0
0.6	4	894	0	6	900	1800	100	100	99	0	1
0.7	4	880	0	20	900	1800	99	100	98	0	2
0.8	4	838	0	62	900	1800	97	100	93	0	7
0.9	4	641	0	259	900	1800	86	100	71	0	29
1	4	0	0	900	900	1800	50	0	0	0	100

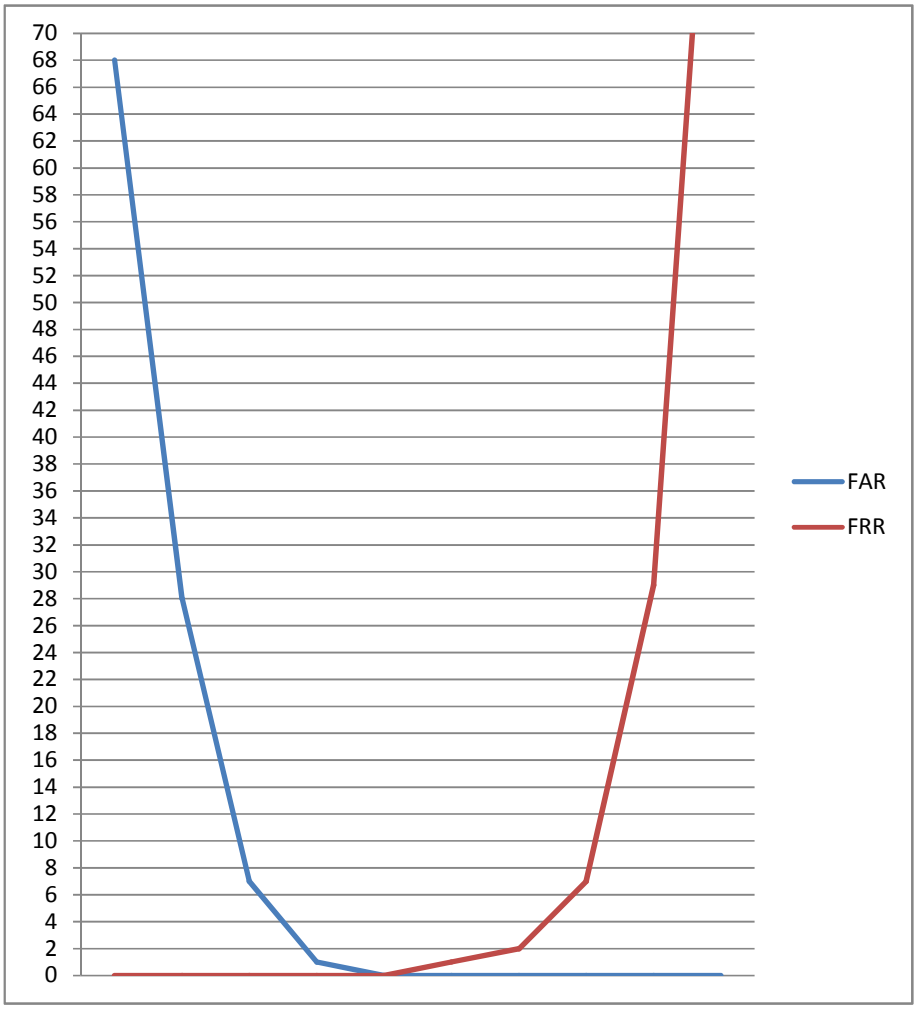

Figure 6.12 EER sketch for the experimental result of the Amharic word code '04'

- Maximum accuracy obtained for Amharic word coded '04' is 100% with 100% precision, 100% recall and 0.00% EER at 0.5 theta value.

6.8.5 SVM Polynomial Kernel result Analysis for the Amharic Word coded '05'

Table 6.15 Experimental result for Amharic word code '05'

Theta	TEXT	TA	FA	FR	TR	TOTAL	ACCURACY	PRECISION	RECALL	FAR	FRR
0	5	900	488	0	412	1800	73	65	100	54	0
0.1	5	900	172	0	728	1800	90	84	100	19	0
0.3	5	900	9	0	891	1800	100	99	100	1	0
0.4	5	900	3	0	897	1800	100	100	100	0	0
0.5	5	900	0	0	900	1800	100	100	100	0	0
0.6	5	897	0	3	900	1800	100	100	100	0	0

0.7	5	891	0	9	900	1800	100	100	99	0	1
0.8	5	859	0	41	900	1800	98	100	95	0	5
0.9	5	727	0	173	900	1800	90	100	81	0	19
1	5	0	0	900	900	1800	50	0	0	0	100

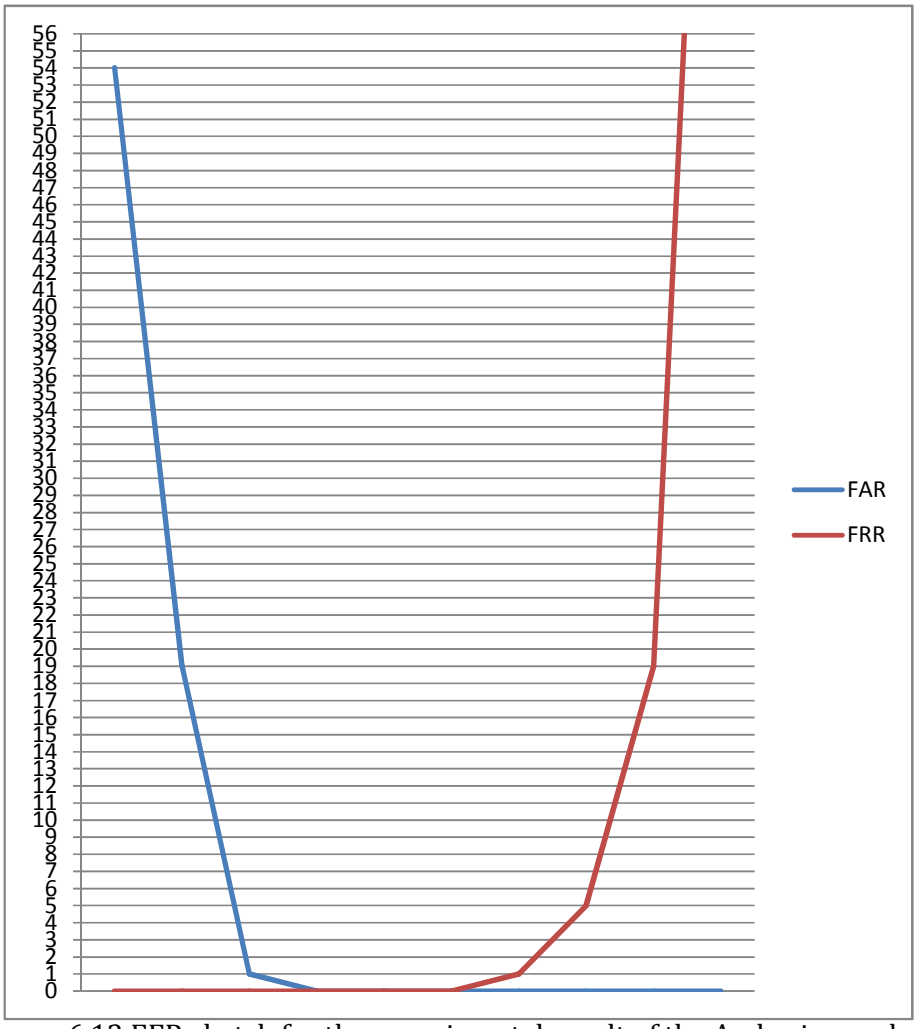

Figure 6.13 EER sketch for the experimental result of the Amharic word code '05'

- Maximum accuracy obtained for Amharic word coded '05' is 100% with 100% precision, 100% recall and 0.00% EER in the range of 0.4-0.6 theta values. Furthermore, at 0.5 theta value, notice also that FA=FR=0.

6.8.6 SVM Polynomial Kernel result Analysis for the Amharic Word coded '06'

Table 6.16 Experimental result for Amharic word code '06'

Theta	TEXT	TA	FA	FR	TR	TOTAL	ACCURACY	PRECISION	RECALL	FAR	FRR
0	6	900	556	0	344	1800	69	62	100	62	0

0.1	6	900	224	0	676	1800	88	80	100	25	0
0.3	6	900	17	0	883	1800	99	98	100	2	0
0.4	6	900	4	0	896	1800	100	100	100	0	0
0.5	6	899	0	1	900	1800	100	100	100	0	0
0.6	6	896	0	4	900	1800	100	100	100	0	0
0.7	6	883	0	17	900	1800	99	100	98	0	2
0.8	6	841	0	59	900	1800	97	100	93	0	7
0.9	6	675	0	225	900	1800	88	100	75	0	25
1	6	0	0	900	900	1800	50	0	0	0	100

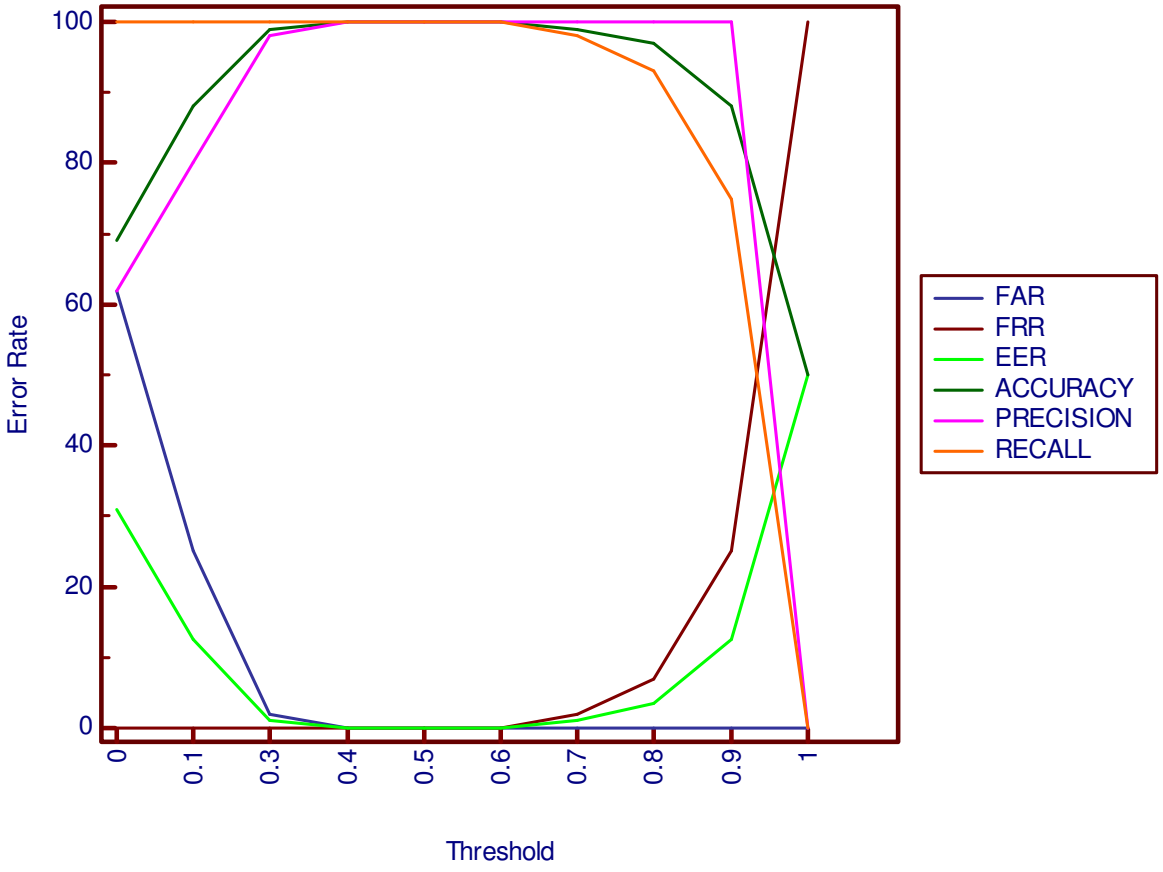

Figure 6.14 Performance sketch for the experimental result of the Amharic word code '06'

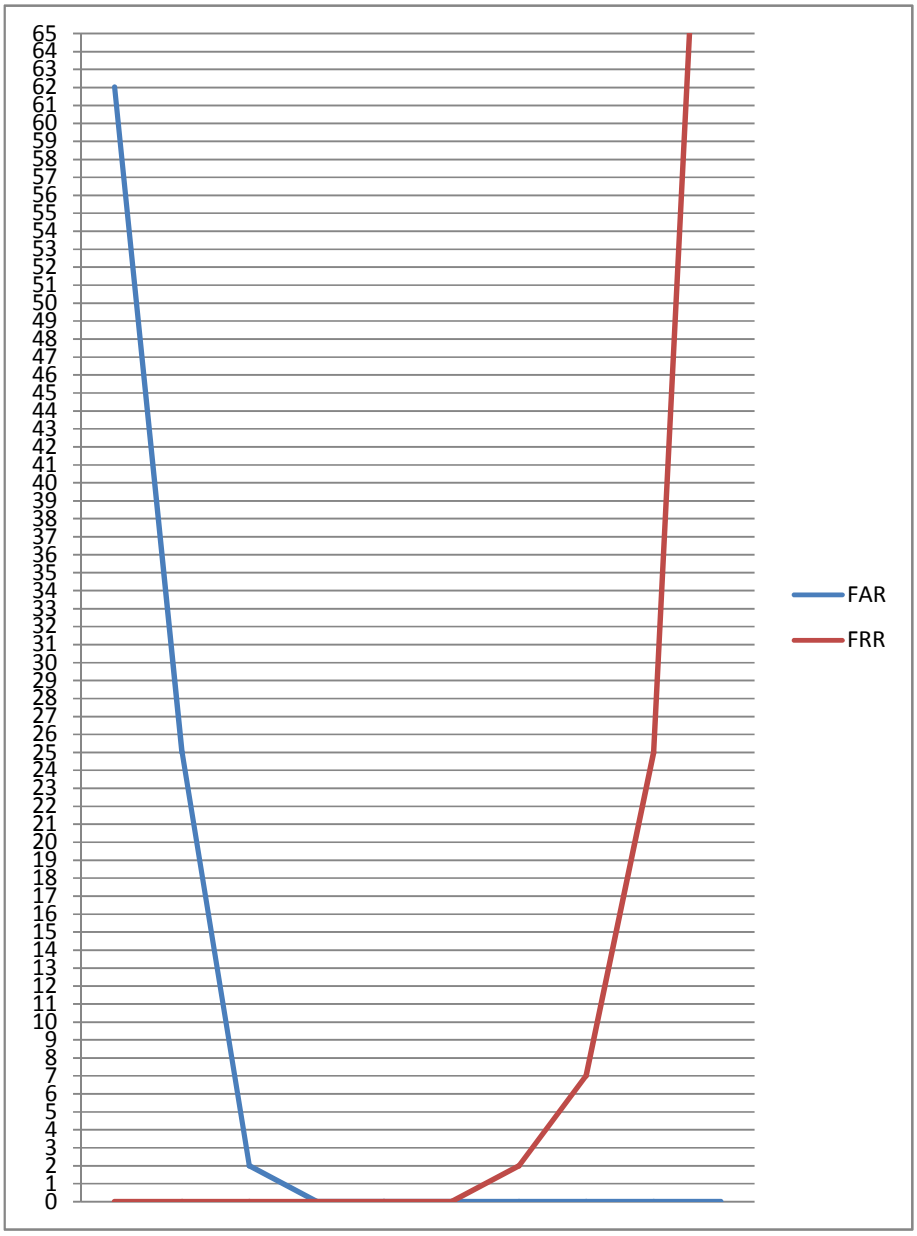

Figure 6.15 EER sketch for the experimental result of the Amharic word code '06'

- Maximum accuracy obtained for Amharic word coded '06' is 100% with 100% precision, 100% recall and 0.00% EER in the range of 0.4-0.6 theta values.

6.8.7 SVM Polynomial Kernel result Analysis for the Amharic Word coded '07'

Table 6.17 Experimental result for Amharic word code '07'

Theta	TEXT	TA	FA	FR	TR	TOTAL	ACCURACY	PRECISION	RECALL	FAR	FRR
0	7	900	536	0	364	1800	70	63	100	60	0
0.1	7	900	189	0	711	1800	90	83	100	21	0
0.3	7	900	14	0	886	1800	99	98	100	2	0
0.4	7	900	4	0	896	1800	100	100	100	0	0
0.5	7	898	1	2	899	1800	100	100	100	0	0
0.6	7	896	0	4	900	1800	100	100	100	0	0
0.7	7	886	0	14	900	1800	99	100	98	0	2
0.8	7	846	0	54	900	1800	97	100	94	0	6
0.9	7	710	0	190	900	1800	89	100	79	0	21
1	7	0	0	900	900	1800	50	0	0	0	100

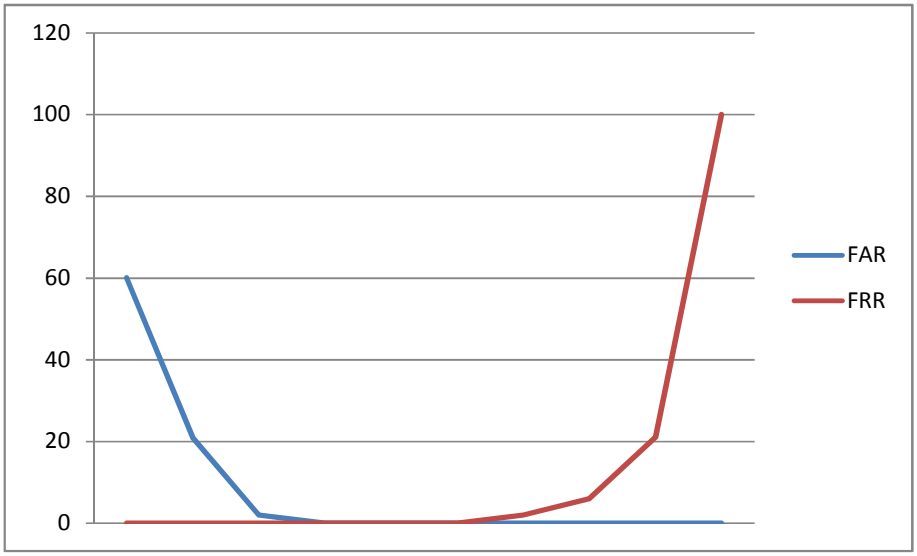

Figure 6.16 EER sketch for the experimental result of the Amharic word code '07'

- Maximum accuracy obtained for Amharic word coded '07' is 100% with 100% precision, 100% recall and 0.00% EER in the range of 0.4-0.6 theta values.

6.8.8 SVM Polynomial Kernel result Analysis for the Amharic Word coded '08'

Table 6.18 Experimental result for Amharic word code '08'

Theta	TEXT	TA	FA	FR	TR	TOTAL	ACCURACY	PRECISION	RECALL	FAR	FRR
0	8	900	431	0	469	1800	76	68	100	48	0
0.1	8	900	117	0	783	1800	94	88	100	13	0
0.3	8	900	3	0	897	1800	100	100	100	0	0
0.4	8	900	1	0	899	1800	100	100	100	0	0
0.5	8	900	0	0	900	1800	100	100	100	0	0
0.6	8	899	0	1	900	1800	100	100	100	0	0
0.7	8	897	0	3	900	1800	100	100	100	0	0
0.8	8	885	0	15	900	1800	99	100	98	0	2
0.9	8	781	0	119	900	1800	93	100	87	0	13
1	8	0	0	900	900	1800	50	0	0	0	100

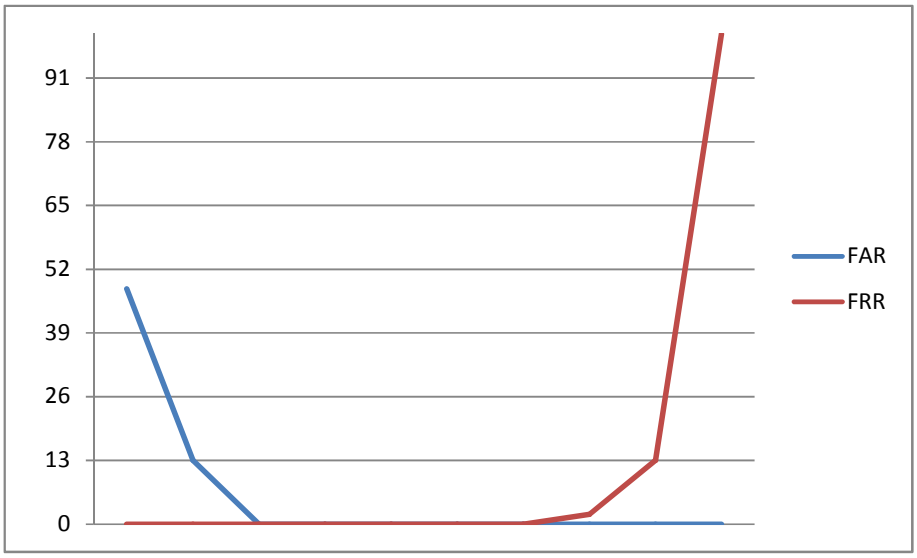

Figure 6.17 EER sketch for the experimental result of the Amharic word code '08'

- Maximum accuracy obtained for Amharic word coded '08' is 100% with 100% precision, 100% recall and 0.00% EER in the range of 0.3-0.7 theta values. At 0.5 theta value, FA=FR=0.

6.8.9 SVM Polynomial Kernel result Analysis for the Amharic Word coded '09'

Table 6.19 Experimental result for Amharic word code '09'

Theta	TEXT	TA	FA	FR	TR	TOTAL	ACCURACY	PRECISION	RECALL	FAR	FRR
0	9	900	474	0	426	1800	74	66	100	53	0
0.1	9	900	132	0	768	1800	93	87	100	15	0
0.3	9	899	12	1	888	1800	99	99	100	1	0
0.4	9	898	8	2	892	1800	99	99	100	1	0
0.5	9	894	4	6	896	1800	99	100	99	0	1
0.6	9	892	2	8	898	1800	99	100	99	0	1
0.7	9	888	1	12	899	1800	99	100	99	0	1
0.8	9	865	0	35	900	1800	98	100	96	0	4
0.9	9	768	0	132	900	1800	93	100	85	0	15
1	9	0	0	900	900	1800	50	0	0	0	100

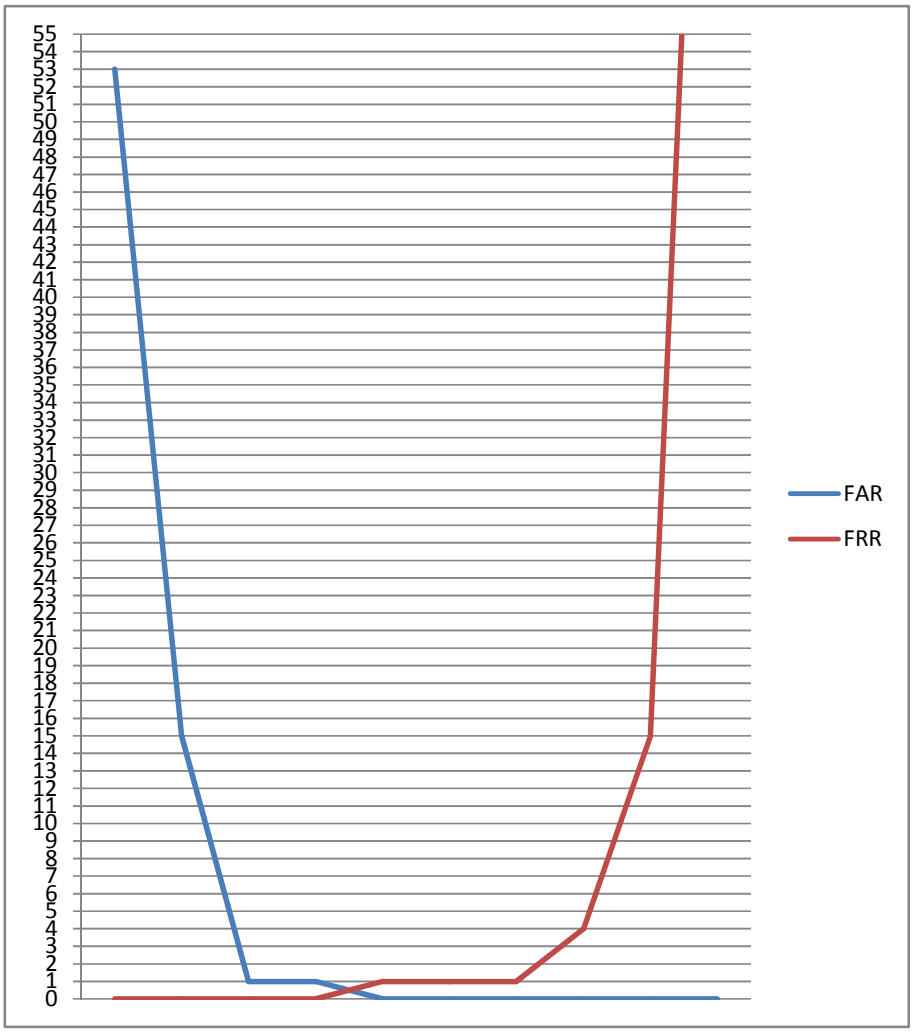

Figure 6.18 EER sketch for the experimental result of the Amharic word code '09'

- Maximum accuracy obtained for Amharic word coded '09' is 99% with 100% precision, 99% recall and 0.50% average EER in the range of 0.5-0.7 theta values. At 0.7 theta value, FA=1 which is the lowest and hence more preferred than 0.5 & 0.6 theta values.

6.8.10 SVM Polynomial Kernel result Analysis for the Amharic Word coded '10'

Table 6.20 Experimental result for Amharic word code '10'

Theta	TEXT	TA	FA	FR	TR	TOTAL	ACCURACY	PRECISION	RECALL	FAR	FRR
0	10	900	490	0	410	1800	73	65	100	54	0
0.1	10	900	156	0	744	1800	91	85	100	17	0
0.3	10	900	16	0	884	1800	99	98	100	2	0
0.4	10	900	5	0	895	1800	100	99	100	1	0
0.5	10	899	1	1	899	1800	100	100	100	0	0
0.6	10	895	0	5	900	1800	100	100	99	0	1
0.7	10	884	0	16	900	1800	99	100	98	0	2
0.8	10	856	0	44	900	1800	98	100	95	0	5
0.9	10	743	0	157	900	1800	91	100	83	0	17
1	10	0	0	900	900	1800	50	0	0	0	100

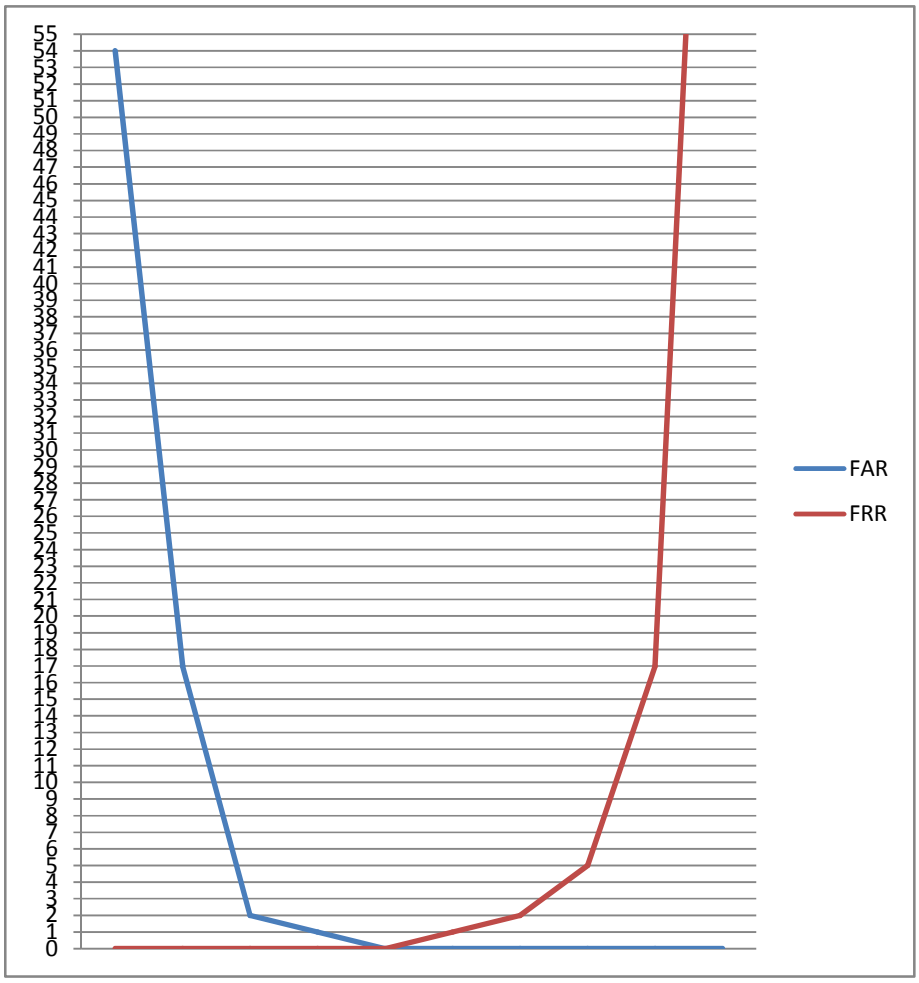

Figure 6.19 EER sketch for the experimental result of the Amharic word code '10'

- Maximum accuracy obtained for Amharic word coded '10' is 100% with 100% precision, 100% recall and 0.00% EER at 0.5 theta value.

6.8.11 Summary for Polynomial Kernel Analysis by Amharic Word

The ATPSVM model performs with 0.00% EER, 100% accuracy, 100% precision and 100% recall for text codes '03', '04', '05', '06', '07', '08', and '10' as reported in Table 6.21.

Table 6.21: Performance summary for each Amharic word

Word code	EER	Accuracy	Precision	Recall
01	1%	99%	99%	99%
02	0.5%	99%	100%	99%
03	0.00%	100%	100%	100%
04	0.00%	100%	100%	100%
05	0.00%	100%	100%	100%
06	0.00%	100%	100%	100%
07	0.00%	100%	100%	100%
08	0.00%	100%	100%	100%
09	0.50%	99%	100%	99%
10	0.00%	100%	100%	100%

The performance of the model for word code '01' is 1% EER where as that of '02' and '09' is average 0.5%. The accuracy, precision, and recall values are each 99% for all these three Amharic words.

- Thus, the discrimination level of the ATPSVM model slightly varies among some Amharic words.

7 CONCLUSION AND FUTURE WORK

7.1 Conclusion

It is noted that using the SVM RBF kernel in one-against-all strategy, [45] has reported 95.1% accuracy with 0.00% EER. The researcher's assumption here is that 95.1% is the average accuracy and the EER 0.00% is taken for a single best performed English word. The feature normalization method applied in [45] is spectral domain followed by dividing each MFCC feature observation by row maximum value.

The logic in [45] is implemented for Amharic and the result obtained is 92.93% average accuracy. The research work for the ATPSVM model is initiated from this implementation and its subsequent poor accuracy performance.

The ATPSVM model applies spectral domain normalization followed by min-max normalization to the MFCCs feature vectors so as to scale the observation feature vector values in [0,1]. Then it applies one-against-each SVM speaker modeling strategy so that to maintain the balance of the test speaker feature vectors in the mixed features.

The ATPSVM model is evaluated using the SVM kernel functions Linear, MLP, RBF and Polynomial. For the SVM linear, the model training does not converge and hence speaker model could not be created. The SVM RBF has resulted better accuracy performance than the SVM MLP kernel. Best accuracy performance is obtained when the SVM polynomial kernel function is applied. This could be attributed to the function definition of the SVM polynomial kernel.

Using the SVM Polynomial kernel for all the 10 Amharic words experimented, the ATPSVM model performs with 99.7% average accuracy, 99.8% average precision, 99.7% average recall and 0.25% EER .

In addition, the ATPSVM model performs with 0.00% EER, 100% accuracy, 100% precision and 100% recall for Amharic words coded '03', '04', '05', '06',

’07’, ‘08’, and ‘10’. For some specific Amharic words coded ‘01’,’02’, and ‘09’, the model performs slightly lower than the rest.

Therefore, the ATPSVM model has shown a promising verification performance better than [45]. This could be attributed to the differentiating techniques applied in the ATPSVM model but not in [45]. The differentiating techniques applied in the ATPSVM model are min-max feature vectors normalization, the SVM polynomial power 3 kernel function, and one-against-each SVM classification strategy.

As to the application of the research work in Ethiopia, the ATPSVM model can be applied so as to safe guard mobile banking service and other secured applications with an appropriate selection of the Amharic word to be prompted. The cut-off points for accuracy and EER so that to ACCEPT/REJECT the speaker can be set depending upon the risk appetite of the service provider organization.

7.2 Future work

The following works could be done next.

- The nature of more discriminative Amharic words so that at all times the performance of the ATPSVM model will be 0.00% EER with 100% accuracy.
- This model can be extended for other Ethiopian Languages.
- The text-independent speaker verification method can be implemented using Amharic & other Ethiopian language speeches.

8 REFERENCES

[1]. Glasser, A**.,** *Fundamentals of Speaker Verification and Biometric-Based Authentication*. Opus Research, Inc., August 2005.

[2]. Furui S., *Recent advances in speaker recognition. Pattern Recognition Letters* 18, 9 (September 1997), 859–872.

[3]. http://www.cs.bham.ac.uk/~mdr/teaching/modules/security/lectures/biometric.html, Date accessed Nov 17, 2010.

[4]. J. Abbazio, S. Perez, D. Silva, R. Tesoriero, F. Penna and R. Zack, *Proc. Student-Faculty Research Day*, CSIS, New York, USA, pp. C1.1-C1.8 (2009).

[5]. http://www.barcode.ro/tutorials/biometrics/voice.html, Date accessed Apr 25, 2011.

[6]. http://www.voicevault.com/casestudies.aspx , last date accessed on Apr 25, 2011.

[7]. J. R. Deller, J. H. L. Hansen, J. G. Proakis, *Discrete-Time Processing of Speech Signals*, Piscataway (N.J.), IEEE Press, 2000.

[8]. John R. Vacca, *Biometric Technologies and Verification Systems*, Elsevier Inc, 2007.

[9]. John Holmes,Wendy Holmes, *Speech Synthesis and Recognition Second Edition*, Taylor & Francis,2001.

[10]. Aitken, C. G. G., Zadora, G., and Lucy, D., "*A definition of biometrics.* ", National Biometric Test Center Collected works, *San Jose State University,* 2000.

[11]. Neil T. Kleynhans, Etienne Barnard, "*Language dependence in multilingual speaker verification*", University of Pretoria.

[12]. Vincent Wan, "*Speaker Verification using support vector machines*", University of Sheffield,2003

[13]. http://www.biobimo.eurecom.fr/publications/GMMHMM.pdf, last date accessed on Apr 25, 2011.

[14]. Tomi Kinnunen, Haizhou Lib, *An Overview of Text-Independent Speaker Recognition: from Features to Supervectors*, July 1, 2009.

[15]. E.M. Weddin, Ole Winther, *Frame Selection for Speaker Identification*, IMM publications, 2005.

[16]. France Mihelic and Janez Žibert, *Speech Recognition, Technologies and Applications*, In-Tech, 2008.

[17]. S.S. Stevens, J. volkman, and E. B. Newman, *A scale for the measurement of the psychological magnitude of pitch*, J. Acoust Soc Amer, 8: 185-19, 1937.

[18]. S. SARASWATHI, R. VISHALAKSHY, *Design of Multilingual Speech Synthesis System*, January 2010.

[19]. S. Furui, "*Digital Speech Processing, Synthesis and Recognition*", New York, Marcel Dekker, 2001.

[20]. B. S. Atal, "*Automatic Recognition of Speakers from their Voices*", Proceedings of the IEEE, vol 64, 1976, pp 460 – 475.

[21]. http://www.lsv.univsarland.de/dsp_ss05_chap9.pdf, last date accessed on Apr 25, 2011.

[22]. D. A. Reynolds, "*Experimental Evaluation of Features for Robust Speaker dentification*", IEEE Transactions on Speech and Audio Processing, Vol. 2, No 4, October 1994, pp. 639-643.

[23]. Campbell, J., *Speaker recognition: a tutorial*. Proceedings of the IEEE 85, 9 (September 1997), 1437–1462.

[24]. BenZeghiba, M., and Bourland, H., *User-customized password speaker verification using multiple reference and background models*. Speech Communication 48, 9 (September 2006), 1200–1213.

[25]. Furui S., *Cepstral analysis technique for automatic speaker verification*. IEEE Transactions on Acoustics, Speech and Signal Processing 29, 2 (April 1981), 254–272.

[26]. Soong, F., A.E., A. R., Juang, B.-H., and Rabiner, L., *A vector quantization approach to speaker recognition*. AT & T Technical Journal 66 (1987), 14–26.

[27]. Reynolds, D., and Rose R., *Robust text-independent speaker identification using Gaussian mixture speaker models*. IEEE Trans. on Speech and Audio Processing 3 (January 1995), 72– 83.

[28]. Farrell, K., Mammone, R., and Assaleh, K., *Speaker recognition using neural networks and conventional classifiers*. IEEE Trans. on Speech and Audio Processing 2, 1 (January 1994), 194–205.

[29]. Campbell, W., Campbell, J., Reynolds, D., Singer, E., and Torres-Carrasquillo, P., *Support vector machines for speaker and language recognition*. Computer Speech and Language 20, 2-3 (April 2006), 210–229.

[30]. Mahmoud I. Abdalla, Hanaa S. Ali, *Wavelet-Based Mel-Frequency Cepstral Coefficients for Speaker Identification using Hidden Markov Models*. TELECOMMUNICATIONS, VOLUME 1, ISSUE 2, MARCH 2010

[31]. T. Matsui and S. Furui, "*Concatenated phoneme models for text-variable speaker recognition*," Proc. ICASSP, pp. II-391-394, 1993.

[32]. Reynolds D., *Speaker identification and verification using Gaussian mixture speaker models*. Speech Communication 17 (August 1995), 91–108.

[33]. Reynolds, D., Quatieri, T., and Dunn, R., *Speaker verification using adapted gaussian mixture models*. Digital Signal Processing 10, 1 (January 2000), 19–41.

[34]. R.V Pawar, P.P.Kajave, and S.N.Mali, *Speaker Identification using Neural Networks*, 2005.

[35]. Bimbot, F., Bonastre, J.-F., Fredouille, C., Gravier, G., Magrin-Chagnolleau, I., Meignier, S., Merlin, T., Ortega-Garcia, J., etrovska-Delacretaz, D., and Reynolds, D., *A tutorial on text-independent speaker verification*. EURASIP Journal on Applied Signal Processing 2004, 4 (2004), 430–451.

[36]. Doru-Petru Munteanu, Stefan-Adrian Toma, "*Automatic Speaker Verification Experiments using HMM*", 2010.

[37]. Cheang Soo Yee and Abdul Manan Ahmad, *Malay Language Text-Independent Speaker verification using NN-MLP classifier With MFCC*, 2008 International Conference on Electronic Design, December 1-3,2008, Penang, Malaysia.

[38]. Mohamed Alkanhal, Mansour Alghamdi and Zeeshan Muzaffar, *Speaker Verification based on Saudi Accented Arabic Database*, 2008.

[39]. Răstoceanu Florin and Diana Militaru, *HMM/SVM Hybrid Method for Speaker Verification*, 2010.

[40]. Javier R. Saeta, Javier Hernando, Oscar Manso, Manel Medina, *Applying Speaker Verification to Certificate Revocation,* 2004: A Speaker Odyssey, The Speaker Recognition Workshop, Toledo, May 2004, p.381-384.

[41]. Dijana Petrovska Delacr´etaz and Jean Hennebert*, Text-prompted speaker verification* experiments with phoneme specific MLPs.

[42]. Chris Longworth, *Kernel Methods for Text-Independent Speaker Verification*, 2010.

[43]. Mohd Zaizu Ilyas, Salina Abdul Samad, Aini Hussain, Khairul Anuar Ishak, *Speaker Verification using Vector Quantization and Hidden Markov Model,* 2007.

[44]. Shi-Huang Chen, Yu-Ren Luo, and Rodrigo Capobianco Guido, *Speaker Verification Using Line Spectrum Frequency, Formant, and Support Vector Machine*, 2009.

[45]. Shi-Huang Chen, Yu-Ren Luo, *Speaker Verification Using MFCC and Support Vector Machine*, 2009.

[46]. C.C. Lin, S.H. Chen, T. K. Truong, and Yukon Chang, "*Audio Classification and Categorization Based on Wavelets and Support Vector Machine*," IEEE Trans. on Speech and Audio Processing, Vol. 13, No. 5, pp. 644-651, Sept. 2005.

[47]. SWM tutorial by Mingyue Tan, UBC, www.iro.umontreal.ca/~pift6080/H09/documents/.../svm_tutorial.ppt, date accessed Apr 25, 2011.

9 ANNEXES

9.1 Annex1: Matlab Code for Extracting Speaker Features

```
function [x,y,h,n,b,g,y2,y3,y4,y5,y6,y7]  = extractSpeakerFeatures()
    %
    %===========Variable stands for===============
    %
    %x is sample wave file
    %y is MFCC feature vector
    %h is MFCC order (feature)
    %n is number of feature observations
    %b is the mean for observations for each feature vector (order)
    %g is the maximum for observations for each feature vector (order)
    %y2 is the value of y after b is subtracted
    %y3 is the value after y2 is divided by g
    %y4 is the transpose of y3. i.e when feature vectors are arranged in rows
    %y5 is the appended set of all y4
    %y6 is the final merged feature vector for training
    %y7 is the final merged feature vector for testing
    %
    %===========Initialization===============
    %
    clc;
    close all;
    clear all; % clears all memory variables
    x0='d:\speechwavefiles\';
    x01='d:\speakerfeatures\';
    %
    %===== extracting Training and Testing Feature vectors separately ====
    %
  for text=1:10 %Anid,Hulet,...,Asir. These are codes for the Amharic words
      t1=int2strconc(text);
      t1=pad(t1); %_01_ For locating speech wave files from disk.
        for test=1:10         %test utterance data  _01_01
          y7=[];
           y5=[];
           y6=[];
           for train=1:10         %train utterance data except _01_01
              for a=1:10 % speakers  01_01_01
                    p1=int2strconc(a);
                    x1=[x0,[p1,t1]];  %01_01_
                    x2=[x1,int2strconc(train)];  %01_01_01
                    x3=[x2,'.wav']; %concatenated address of a wave file
                    x=wavread(x3); % read the speech wave file
                    y=mfcc(x); %extract MFCC feature vectors from the file
                    x3=[x2,'.mfcc'];
                    save(x3,'y'); % save MFCC feature vectors
                    [h n]=size(y); % number of rows (h) and columns (n) of y
                    y2=zeros(h,n);
                    y3=zeros(h,n);
                    %
                    %==Finding Mean for observations for each Feature vector=
                    %
                    for i=1:h; %for each row
                        mean=0;
                        for j=1:n;
                            mean=mean+(y(i,j)/n);
                         end;
                            b(i)=mean; %mean
                      end;
```

```
                    %=====Subtracting the Mean from observations ==
                    %
                    for k=1:h;
                        for j=1:n;
                            y2(k,j)=y(k,j)-b(k);%mean subtracted
                        end;
                    end;
                    %
                    %===Finding Max and min observation values for each feature
                    %
                    for m=1:h;
                        g(m)=y2(m,1);
                        min(m)=y2(m,1);
                        for q=2:n;
                            if y2(m,q) >g(m);
                               g(m)=y2(m,q); % row maximum value obtained.
                            end;
                            if y2(m,q) < min(m);
                                min(m)=y2(m,q); % row min value obtained.
                            end;
                         end;
                    end;
                    %
                    %=========Normalize observations for each Feature vector
                    %
                    for s=1:h;
                        diff=g(s)-min(s);
                        for z=1:n;
                            y3(s,z)=(y2(s,z)-min(s))/diff;%get values in [0,1]
                        end;
                    end;
                    %
                    %==Transposing the data as observations vs Feature vectors ==
                    %
                    y4=y3'; % transposition
                    %
                    %=====Labelling the owner of the speech ============
                    %
                    [f,v]=size(y4);
                    r=a*ones(f,1); % for speaker a
                    y4=[r y4]; % concatenate speaker ID with features
                    %
                    %===Extracting testing data as y7 and training data as y5
                    %
                    if train == test
                       y7=[y7;y4];  % test data set appended
                    else
                       y5=[y5;y4]; % training data set y4 appended to y5.
                    end
                end % speaker  01_01_01
            end  %train utterance data except _01_01
            x7=[x01,['text',[t1,['test_',[int2strconc(test),'.xls']]]]];
            xlswrite(x7,y7); % save training data set to a single Excel file.
            x6=[x01,['text',[t1,['test_',[int2strconc(test),'_train.xls']]]]];
            xlswrite(x6,y5); % save testing data set to another Excel file.
        end %for test data
    end % for text
    %===========End of extractSpeakerFeatures() ========================
```

9.2 Annex 2: Matlab Code for SVM Functionalities

```
Function [datatrain,grouptrain,datatest,grouptest,data2,data3,classes]=svm()
    %
    %===========Variable stands for===============
    %
    %datatrain contains feature vectors extracted for training
    %datatest contains feature vectors extracted for testing
    %grouptrain contains group labels for feature vectors to be used for training
    %grouptest contains group labels for feature vectors to be used for testing
    %data2 is labelled training feature vector
    %data3 is labelled testing feature vector
    %classes to mean labels
    %svmStruct contains structure generated by svmtrain
    %
    %===========Initialization================
    %
    clc;
    close all;
    clear all; % clears all memory variables
    v2=[]; % initialize
    for i=0:10 % the 11 values of threshold theta
        theta = 0.1;
        theta=theta*i;
        trueRejection_t = 0;
        falseAcceptance_t = 0;
        falseRejection_t = 0;
        trueAcceptance_t = 0;
        x01='d:\speakerfeatures\';
        disp(sprintf('Performance at thetha = %f',theta));
        disp(sprintf('Text TA FA FR TR Total accuracy  precision  recall FAR FRR'));
        for text=1:10          %anid,hulet,sost,..,asir.
            trueRejection = 0;
            falseAcceptance = 0;
            falseRejection = 0;
            trueAcceptance = 0;
            t1=int2strconc(text);
            t1=pad(t1); %_01_
            for test=1:10          %test utterance data  _01_01
                x6=[x01,['text',[t1,['test_',[int2strconc(test),'_train.xls']]]]];
                x7=[x01,['text',[t1,['test_',[int2strconc(test),'.xls']]]]];
                datatrain=xlsread(x6);%b.ans;
                datatest=xlsread(x7);
                for speaker =1:10
                   for otherSpeaker = 1:10
                      if(otherSpeaker ~= speaker)
                        datatrain2 = [];
                        [len, dim] = size(datatrain);
                        for k=1:len
                          if (datatrain(k,1)==speaker || datatrain(k,1)== otherSpeaker)
                             datatrain2 = [datatrain2;datatrain(k,:)];
                          end
                        end
                        grouptrain = ismember(datatrain2(:,1),speaker);%Groups
                        data2=datatrain2;
                        data2(:,1)=[];
                        %
                        %=====Kernel function alternatives=======
                        %
                    %k2= svmtrain(data2,grouptrain,'Kernel_Function','rbf','RBF_Sigma',1);
```

```
                    k2= svmtrain(data2,grouptrain,'Kernel_Function','polynomial','polyorder',3);
                    %k2= svmtrain(data2,grouptrain,'Kernel_Function','mlp','mlp_params',[1, -1]);
                    %========testing==================================
                    %
                    %select from datatest whose class label is speaker
                    [lengthTotal, dim] = size(datatest);
                    for testSpeaker =[speaker, otherSpeaker]
                        startRow = 0;
                        endRow = 0;
                        for rowindex =1: lengthTotal
                            if(startRow== 0 && datatest(rowindex,1)==testSpeaker)
                                startRow = rowindex;
                            end
                            if(startRow~= 0 && datatest(rowindex,1)~= testSpeaker)
                                endRow = rowindex-1;
                            end
                            if (endRow ~= 0)
                                break
                            end
                        end
                        if (endRow == 0)
                            endRow = lengthTotal;
                        end
                        selectedData =datatest(startRow:endRow,:);
                        selectedData(:,1)=[];
                        classes = svmclassify(k2,selectedData); %classifier
                        %
                        %=========Speaker verification Decision Making====
                        %
                        [lengthTest, dim] = size(classes);
                        total_count_1=0;
                        total_count_0=0;
                        for i=1:lengthTest
                            if(classes(i,1)==1)
                                total_count_1 = total_count_1 + 1;
                            else
                                total_count_0   = total_count_0 + 1;
                            end
                        end
                        %
                        %========= Performance Measurement===========
                        %
                        performance=total_count_1/(total_count_1+ total_count_0);
                        if performance > theta % speaker accepted
                            if(speaker ==testSpeaker)
                                trueAcceptance = trueAcceptance + 1;
                            else
                                falseAcceptance = falseAcceptance + 1;
                            end
                         else % speaker rejected
                            if(speaker ==testSpeaker)
                                falseRejection = falseRejection + 1;
                            else
                                trueRejection = trueRejection + 1;
                            end
                        end
                    end %Test speaker
                end %if
            end %other speaker
        end %speaker
    end %test utterance
```

```
    total = trueAcceptance + falseAcceptance + falseRejection + trueRejection;
    %====report the performance based on the text
    accuracy=round(100*(trueAcceptance+trueRejection)/total) ;
    precision=round(100*(trueAcceptance)/(trueAcceptance+falseAcceptance)) ;
    recall=  round(100*(trueAcceptance)/(trueAcceptance+falseRejection)) ;
    far  =round(100*(falseAcceptance)/(falseAcceptance+trueRejection)) ;
    frr=round(100*(falseRejection)/(falseRejection+trueAcceptance)) ;
    result = sprintf('%d            %d        %d        %d        %d
                %d        %d        %d        %d      %d       %d\n',text,
                trueAcceptance,falseAcceptance,falseRejection,trueRejection
                ,total,accuracy,  precision,  recall,  far,  frr);
    disp(result);
    v=[text trueAcceptance falseAcceptance falseRejection trueRejection
      total accuracy precision recall far frr  theta];
    v2=[v2;v]; % to append the output
    %===sum up to get the aggregate performance
    trueAcceptance_t = trueAcceptance_t + trueAcceptance;
    falseAcceptance_t = falseAcceptance_t + falseAcceptance;
    falseRejection_t= falseRejection_t + falseRejection;
    trueRejection_t = trueRejection_t + trueRejection;
    total_t = trueAcceptance_t+ falseAcceptance_t + falseRejection_t + trueRejection_t;
 end  % text
  result = sprintf('Total %d          %d        %d        %d    %d\n'
     ,trueAcceptance_t,falseAcceptance_t,falseRejection_t,trueRejection_t,total_t);
  disp(result);
end %theta
xlswrite([x01,'Polynomial_kernel_result.xls'],v2);
%
%=========End of svm() function=======================================
```

9.3 Annex 3: Matlab Code adopted & customized for MFCCs calculation

```
function [ceps,freqresp,fb,fbrecon,freqrecon] = mfcc(input, samplingRate, frameRate)

%  mfcc - Mel frequency cepstrum coefficient analysis.
%   [ceps,freqresp,fb,fbrecon,freqrecon] = ...
%           mfcc(input, samplingRate, [frameRate])
% Find the cepstral coefficients (ceps) corresponding to the
% input.  Four other quantities are optionally returned that
% represent:
%   the detailed fft magnitude (freqresp) used in MFCC calculation,
%   the mel-scale filter bank output (fb)
%   the filter bank output by inverting the cepstrals with a cosine
%       transform (fbrecon),
%   the smooth frequency response by interpolating the fb reconstruction
%       (freqrecon)
%  -- Malcolm Slaney, August 1993
% Modified a bit to make testing an algorithm easier... 4/15/94
% Fixed Cosine Transform (indices of cos() were swapped) - 5/26/95
% Added optional frameRate argument - 6/8/95
% Added proper filterbank reconstruction using inverse DCT - 10/27/95
% Added filterbank inversion to reconstruct spectrum - 11/1/95

% (c) 1998 Interval Research Corporation

global mfccDCTMatrix mfccFilterWeights

[r c] = size(input);
if (r > c)
    input=input';
end

%   Filter bank parameters
lowestFrequency = 133.3333;
linearFilters = 13;
linearSpacing = 66.66666666;
logFilters = 27;
logSpacing = 1.0711703;
fftSize = 512;
cepstralCoefficients = 22;
windowSize = 512;
%windowSize = 256;      % Standard says 400, but 256 makes more sense
                % Really should be a function of the sample
                % rate (and the lowestFrequency) and the
                % frame rate.
if (nargin < 2) samplingRate = 16000; end;
if (nargin < 3) frameRate = 40; end;
%frameRate
% Keep this around for later....
totalFilters = linearFilters + logFilters;

% Now figure the band edges.  Interesting frequencies are spaced
% by linearSpacing for a while, then go logarithmic.  First figure
% all the interesting frequencies.  Lower, center, and upper band
% edges are all consequtive interesting frequencies.

freqs = lowestFrequency + (0:linearFilters-1)*linearSpacing;
freqs(linearFilters+1:totalFilters+2) = ...
```

```
            freqs(linearFilters) * logSpacing.^(1:logFilters+2);

lower = freqs(1:totalFilters);
center = freqs(2:totalFilters+1);
upper = freqs(3:totalFilters+2);

% We now want to combine FFT bins so that each filter has unit
% weight, assuming a triangular weighting function.  First figure
% out the height of the triangle, then we can figure out each
% frequencies contribution
mfccFilterWeights = zeros(totalFilters,fftSize);
triangleHeight = 2./(upper-lower);
fftFreqs = (0:fftSize-1)/fftSize*samplingRate;

for chan=1:totalFilters
    mfccFilterWeights(chan,:) = ...
  (fftFreqs > lower(chan) & fftFreqs <= center(chan)).* ...
   triangleHeight(chan).*(fftFreqs-lower(chan))/(center(chan)-lower(chan)) + ...
  (fftFreqs > center(chan) & fftFreqs < upper(chan)).* ...
   triangleHeight(chan).*(upper(chan)-fftFreqs)/(upper(chan)-center(chan));
end
%semilogx(fftFreqs,mfccFilterWeights')
%axis([lower(1) upper(totalFilters) 0 max(max(mfccFilterWeights))])

hamWindow = 0.54 - 0.46*cos(2*pi*(0:windowSize-1)/windowSize);

if 0                    % Window it like ComplexSpectrum
    windowStep = samplingRate/frameRate;
    a = .54;
    b = -.46;
    wr = sqrt(windowStep/windowSize);
    phi = pi/windowSize;
    hamWindow = 2*wr/sqrt(4*a*a+2*b*b)* ...
        (a + b*cos(2*pi*(0:windowSize-1)/windowSize + phi));
end

% Figure out Discrete Cosine Transform.  We want a matrix
% dct(i,j) which is totalFilters x cepstralCoefficients in size.
% The i,j component is given by
%                 cos( i * (j+0.5)/totalFilters pi )
% where we have assumed that i and j start at 0.
mfccDCTMatrix = 1/sqrt(totalFilters/2)*cos((0:(cepstralCoefficients-1))' * ...
                (2*(0:(totalFilters-1))+1) * pi/2/totalFilters);
mfccDCTMatrix(1,:) = mfccDCTMatrix(1,:) * sqrt(2)/2;

%imagesc(mfccDCTMatrix);
% Filter the input with the preemphasis filter.  Also figure how
% many columns of data we will end up with.
if 1
    preEmphasized = filter([1 -.97], 1, input);
else
    preEmphasized = input;
end
windowStep = samplingRate/frameRate;
cols = fix((length(input)-windowSize)/windowStep);

% Allocate all the space we need for the output arrays.
ceps = zeros(cepstralCoefficients, cols);
if (nargout > 1) freqresp = zeros(fftSize/2, cols); end;
if (nargout > 2) fb = zeros(totalFilters, cols); end;
```

```
% Invert the filter bank center frequencies.  For each FFT bin
% we want to know the exact position in the filter bank to find
% the original frequency response.  The next block of code finds the
% integer and fractional sampling positions.
if (nargout > 4)
    fr = (0:(fftSize/2-1))'/(fftSize/2)*samplingRate/2;
    j = 1;
    for i=1:(fftSize/2)
        if fr(i) > center(j+1)
            j = j + 1;
        end
        if j > totalFilters-1
            j = totalFilters-1;
        end
        fr(i) = min(totalFilters-.0001, ...
            max(1,j + (fr(i)-center(j))/(center(j+1)-center(j))));
    end
    fri = fix(fr);
    frac = fr - fri;

    freqrecon = zeros(fftSize/2, cols);
end
% Ok, now let's do the processing.  For each chunk of data:
%    * Window the data with a hamming window,
%    * Shift it into FFT order,
%    * Find the magnitude of the fft,
%    * Convert the fft data into filter bank outputs,
%    * Find the log base 10,
%    * Find the cosine transform to reduce dimensionality.
for start=0:cols-1
    first = floor(start*windowStep) + 1;
    last = first + windowSize-1;
    fftData = zeros(1,fftSize);
    fftData(1:windowSize) = preEmphasized(first:last).*hamWindow;
    fftMag = abs(fft(fftData));
    earMag = log10(mfccFilterWeights * fftMag');

    ceps(:,start+1) = mfccDCTMatrix * earMag;
    if (nargout > 1) freqresp(:,start+1) = fftMag(1:fftSize/2)'; end;
    if (nargout > 2) fb(:,start+1) = earMag; end
    if (nargout > 3)
        fbrecon(:,start+1) = ...
            mfccDCTMatrix(1:cepstralCoefficients,:)' * ...
            ceps(:,start+1);
    end
    if (nargout > 4)
        f10 = 10.^fbrecon(:,start+1);
        freqrecon(:,start+1) = samplingRate/fftSize * ...
            (f10(fri).*(1-frac) + f10(fri+1).*frac);
    end
end
% OK, just to check things, let's also reconstruct the original FB
% output.  We do this by multiplying the cepstral data by the transpose
% of the original DCT matrix.  This all works because we were careful to
% scale the DCT matrix so it was orthonormal.
if 1 & (nargout > 3)
    fbrecon = mfccDCTMatrix(1:cepstralCoefficients,:)' * ceps;
%   imagesc(mt(:,1:cepstralCoefficients)*mfccDCTMatrix);
end;

%***************** END of mfcc() function *****************************
```

10 DECLARATION

I declare that this thesis is composed by myself, the work contained herein is my own except where explicitly stated otherwise in the text, and it has not been submitted for any other degree or professional qualifications except specified.

Wolelaw Berhane Alemu

_____

May 16, 2011

The thesis has been submitted for examination with my approval as university advisor.

_____

Sebsibe Hailemariam (PhD)