Authentication: From Passwords to Biometrics

An implementation of a speaker recognition system on Android

Erlend Heimark

Master of Science in Communication Technology
Submission date: June 2012
Supervisor: Danilo Gligoroski, ITEM
Co-supervisor: Yanling Chen, ITEM

Norwegian University of Science and Technology
Department of Telematics

Problem Description

Name of student: Erlend Heimark

Nowadays portable devices such as smart phones are widely used. They often have access to Telecom networks and to the Internet. This makes it possible to have an immediate access to many services such as voice mail or telephone banking anytime and anywhere. Many of these services involve user authentication that currently is done usually by using PIN or passwords.

Biometric systems have a great potential to be used for reliable user authentication. In particular, we note that most portable devices are equipped with a microphone and a video camera. So it is natural to combine with these functionalities to build a friendly and personalized authentication scheme based on face or voice recognition or both.

In this research work, an authentication application for the Android platform is to be made. The application shall use voice recognition as the authentication method. A theoretical study on such a biometric system shall be conducted to have insights into the possible information leakage of sensitive personal data stored. Testing and evaluation of the application shall be performed as well.

For this thesis project, the student has to have programming skills (Java, C/C++), ability to analyze, and willingness to perform original research.

Assignment given: January 16, 2012

Professor: Danilo Gligoroski

Supervisor: Yanling Chen

Abstract

We implement a biometric authentication system on the Android platform, which is based on text-dependent speaker recognition. The Android version used in the application is Android 4.0. The application makes use of the Modular Audio Recognition Framework, from which many of the algorithms are adapted in the processes of preprocessing and feature extraction. In addition, we employ the Dynamic Time Warping (DTW) algorithm for the comparison of different voice features. A training procedure is implemented, using the DTW algorithm to align features. Furthermore, we introduce personal thresholds, based on which the performance for each individual user can be further optimized.

We have carried out several tests in order to evaluate the performance of the developed system. The tests are performed on 16 persons, with in total 240 voice samples, of which 15 samples are from each person. As a result, for authentication, one of the optimal trade-offs of the False Acceptance Rate (FAR) and False Rejection Rate (FRR) achieved by the system is shown to be 13% and 12%, respectively. For identification, the system could identify the user correctly with a rate of 81%. Our results show that one can actually improve the system performance in terms of FAR and FRR significantly, through using the training procedure and the personal thresholds.

Sammendrag

En tekst-avhengig stemmegjenkjennings-applikasjon har blitt laget til Android-plattformen. Versjonen av Android brukt er Android 4.0. Applikasjonen bruker Modular Audio Recognition Framework for mange av algoritmene brukt for preprosessering og egenskapsuttrekking. Dynamic Time Warping-algoritmen (DTW) har blitt implementert for bruk i sammenligningen av disse egenskapene. En treningsprosedyre er også implementert. Prosedyren bruker DTW-algoritmen for å korrigere forskjeller mellom egenskapene. Personlige terskler er innført for å muliggjøre optimalisering av ytelsen for hver enkelt bruker.

Flere tester er utført for å evaluere systemets ytelse. Testene er utført på 16 personer, med totalt 240 opptak, hvor hver person har bidratt med 15 opptak. I forhold til autentisering er en av de optimale trade-offene for ratene for falsk akseptanse og falsk avvisning beregnet til å være henholdsvis 13 og 12 prosent. I forhold til identifisering er systemets identifikasjonsrate beregnet til 81%. Resultatene viser at bruken av treningsprosedyren og personlige terskler, forbedrer ytelsen betydelig.

Preface

The work in this report has been carried out during the spring semester 2012 at the Norwegian University of Science and Technology (NTNU), Institute of Telematics (ITEM). The report is the final documentation of a master thesis in information security.

The author would like to thank friends and family for help and support during the semester the work has been performed. The author would also like to thank Professor Danilo Gligoroski and supervisor Yanling Chen for important and valuable contributions throughout the semester.

Best regards,

Erlend Heimark
June 18, 2012

Contents

List of Figures

List of Tables

Acronyms

API	Application Programming Interface
DTW	Dynamic Time Warping
FAR	False Acceptance Rate
FRR	False Rejection Rate
FMR	False Match Rate
FNMR	False Non-Match Rate
LPC	Linear Predictive Coding
MARF	Modular Audio Recognition Framework
OS	Operating System
SDK	System Developer's Kit

Chapter 1

Introduction

The use of smartphones has become more and more popular over the last years. A study from the market research firm Nielsen [1] shows that the amount of mobile subscribers in America that has a smartphone, has increased from 38% to 50% only in the last year. The introduction of smartphones has made it easier to use many services, such as online banking, anytime and anywhere. Many of these different services have a need for authentication of the user. The typical authentication method in use today is password-based authentication.

Biometrics is considered as a very good method for authentication because of its tight connection to the user, and its convenience. Biometric characteristics are physically connected to a person, making them very reliable for both identification and authentication. With the increase in computational power on mobile devices such as smartphones smartphones, it is now feasible to use biometrics on such devices.

An authentication application using voice biometrics has been made for the Android mobile platform. The application uses speaker recognition as the authentication method. This thesis will describe the implemented application, and evaluate its performance.

1.1 Related work

There exist open-source speaker recognition systems that are built for use on computers. An example is the Modular Audio Recognition Framework [2] that is partially used in the implementation of our application.

To our knowledge, there are no other open-source speaker recognition applications that have been made for the Android platform. There exist some commercial Application Programming Interfaces (API) that are made for enabling the development of speaker recognition systems on mobile devices. Two examples of such API's are the VoiceVault API [3] and the VeriSpeak Embedded Software Developer's Kit (SDK) provided by Neurotechnology [4].

1.2 Thesis outline

The thesis is outlined as follows.

Chapter 2 - Background The chapter provides background information on biometrics and biometric systems. It also gives a description of the different considerations that need to be taken into account regarding security and privacy in such systems. In addition, the chapter gives an introduction to the Android platform.

Chapter 3 - System Architecture The chapter gives an explanation of the architecture of the implemented system. The different algorithms used in the system will also be described in detail.

Chapter 4 - Technical Procedure The chapter describes the several implementations done in the system. The implementation of the training procedure is explained, as well as the introduction of personal thresholds.

Chapter 5 - Experimental Procedure The chapter describes the different experimenting and testing performed on the system.

Chapter 6 - Performance Evaluation The chapter presents the results from the tests described in Chapter 5. These results are then analyzed and discussed with regards to performance of the system.

Chapter 7 - Conclusion and Future Work The chapter summarizes the most important results found, and gives a conclusion on the work done in the thesis. Possible future work that can further improve the performance of the application is also presented.

Chapter 2

Background

This chapter gives an overview of biometrics and biometric systems. The following sections describe biometrics and biometric systems, as well as the security and privacy considerations of such systems. In addition, we also provide a description of the Android platform.

2.1 Biometrics

Biometrics refers to techniques used to recognize individuals by their biological characteristics. Biological characteristics can be viewed as either physiological characteristics or behavioral characteristics. Physiological characteristics are the features of a person that are fixed, meaning something the person is. These characteristics can also be described as passive biological characteristics. Some examples are:

- **Fingerprints:** The analysis of the patterns on a fingertip is a very commonly used method in biometrics.
- **Facial structure:** Facial recognition identifies people by analyzing the different structures in an individual's face.
- **Iris:** Iris recognition analyzes the different patterns of the iris.

Behavioral characteristics regard what a person does. Such characteristics can also be described as active characteristics. Some examples are:

- **Gait:** It is possible to identify people by analyzing their walking pattern.
- **Signature:** Most people have a distinct way they write their signature, which can be used as a behavioral characteristic.

- **Keystroke dynamics:** It is also typical that persons have a distinct way in how they type on a computer keyboard.

Biological characteristics are tightly coupled to a person, making them very suitable for use in authentication. Not all characteristics are equally suited for use in authentication though. Some can be difficult to measure and some can be hard to separate from person to person. Therefore not every characteristic can be used. As stated in [5], four requirements for biological characteristics that are to be used in biometrics are:

- **Universality:** Every person in the world must have this characteristic.
- **Distinctiveness:** The characteristic should be distinctly different for each person.
- **Permanence:** The characteristic should not drastically change for a person over time.
- **Collectability:** It should be easy to measure the characteristic.

The importance of each of these requirements will typically vary, depending on the purpose of the system using these characteristics. If the system has a need of a very high level of security, then universality and distinctiveness becomes very important. If the system focuses more on usability, then collectability becomes more important.

2.2 Authentication and Identification Principles

The definition of authentication varies with regards to what type of environment it applies to. In the implemented biometric system, it is restricted to individuals putting forward an identity claim. The definition will then be as follows: Authentication is the process of verifying that a claimed identity put forward by an individual is legitimate.

To be able to verify such a claim, the individual needs some way of proving his identity. The typical way of doing this is to challenge the individual to provide some kind of information that only the correct person should be in possession of. There are typically three different types of such information:

- Something the individual knows
- Something the individual has
- Something the individual is

Something the individual knows, is some knowledge only the correct person should know about. Typical examples of such information is passwords or questions only the correct person knows the answer to. Something the individual has, are physical objects that only the correct person should be in possession of, and that no one else should be able to get a hold of. An example of such an object is a passport. Something the individual is, relates to biometric characteristics.

The task of the authentication process is to verify that the individual is in possession of such information. In a biometric system this means that the individual needs to provide his biometric characteristics, and the system will verify if these characteristics correspond to the stored characteristics of the claimed identity. The authentication process can be viewed as the same as a verification process. It is a one-to-one comparison where we only compare the incoming features with the stored features of the claimed identity. Whenever authentication is mentioned later in the report, this is the process referred to.

The use of biometric information often makes us able to identify a person as well. This is possible to do since biometric characteristics in most cases are unique for a person. In an identification process the individual only puts forward his biometric information. It will then be the system's task to find the correct identity corresponding to the provided biometric characteristics. The system will then need to search among all the identities stored in the system. This can be viewed as a one-to-many comparison.

2.3 Biometric Systems

Figure 2.1 shows the architecture and operations of a typical biometric system. The figure is taken from the ISO Standard, Biometric Information Protection [6].

2.3.1 Subsystems

A biometric system typically consists of five subsystems.

Data Capture Subsystem

The data capture subsystem has the task of capturing the biometric characteristics of an individual. It typically consists of sensors able to capture biometric information. In the case of voice this is typically a microphone able to record the voice.

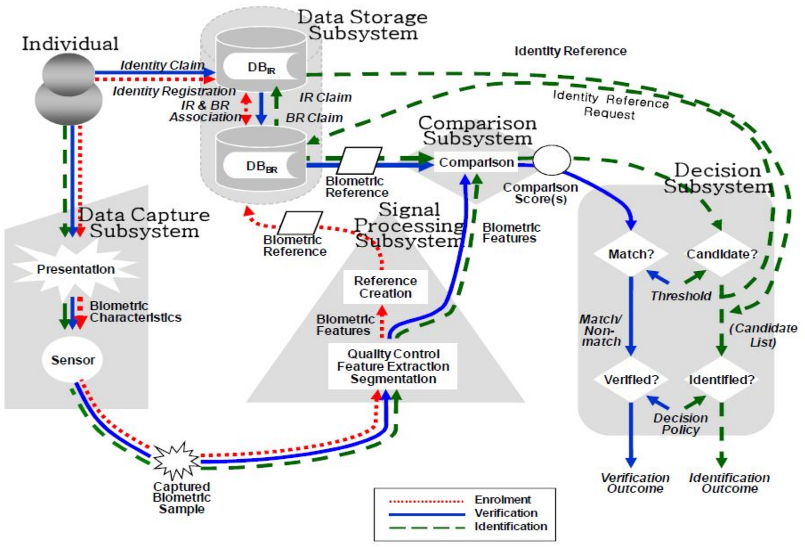

Figure 2.1: Architecture of a biometric authentication system

Signal Processing System

The signal processing subsystem receives the recorded data from the data capture subsystem. Its task is to process the biometric data and extract its important features. The recorded biometric sample should be discarded after this stage, and only the features should be used further on.

Data Storage System

This is the subsystem that handles stored data. It stores the identity of a user and its corresponding biometric reference. A biometric reference contains the biometric features of the user.

Comparison Subsystem

The comparison subsystem is used for comparing biometric features. It computes a comparison score that is further used by the decision subsystem to determine the outcome of the verification or identification process.

Decision Subsystem

The decision subsystem has the task of determining if a user is verified or identified, depending on which of the two cases is used. The subsystem takes as input the calculated score from the comparison. The score is compared against a predetermined threshold. The process of this subsystem is different depending on whether verification or identification is used. In the case of verification, only one comparison score is calculated, and the decision subsystem only needs to check if this score is within the threshold. If it is, the user is verified. In identification, the incoming references need to be compared against all the enrolled references. The decision subsystem then needs to make a list of references that is below this threshold when compared with the incoming reference. A decision is then taken regarding which of the references in the calculated list that is the correct one.

2.3.2 Processes

As we see from Figure 2.1, there are three different processes that can occur in such a system.

Enrolment

Before the system can be used for verification or identification, all users need to be enrolled. This means that all users must provide their identity and biometric characteristics to the system. These are later used as the references during verification or identification. A typical enrolment process is shown by the dotted red line in Figure 2.1. The individual first registers his identity in the system. Then the biometric characteristics are provided. The biometric features are extracted, and then stored as the biometric reference corresponding to the given ID.

Verification

In verification, an individual makes an identity claim against the system, and then provides his biometric information. A typical verification process is shown by the blue line in the figure. The biometric information is recorded and processed by the data capture subsystem. The processed data is sent further on to the signal processing subsystem. Here the features are extracted from the biometric data. The features are then sent to the comparison subsystem. The data storage subsystem checks the claimed identity, and finds its corresponding biometric reference. The reference is then used for comparison. The comparison subsystem takes as input the reference and the incoming features, and calculates a comparison score. This score is used to determine if the individual can be verified as the claimed identity.

Identification

In this case, the individual does not claim his identity; he only provides his biometric information. The task of the system is then to find an enrolled user that has matching biometric characteristics. The difference from the verification case will then be that the system needs to compare each enrolled biometric reference with the incoming reference. The process is shown in Figure 2.1 by the dotted green line. Each comparison score is sent to the decision subsystem. If the score is within the threshold, the subsystem will ask the data storage subsystem for the ID of the compared user. This ID will then be added to a candidate list. When all enrolled references have been checked, the list will consist of all the ID's that had comparison scores within the threshold. The subsystem then takes a decision on which of the ID's in the candidate list that is the correct one.

2.4 Speaker recognition

Speaker recognition is one of several branches of different voice biometrics, or speech processing techniques. Figure 2.2, taken from [7], shows the different branches, as well as the different categories of speaker recognition. The goal of speaker recognition is to recognize who is speaking. This is in contrast to for example speech recognition where the task is to figure out what is spoken.

The two important branches of speaker recognition are speaker identification and speaker verification. Speaker identification is the case where a person inputs a voice recording in the system without an identity claim. The system must then try to identify who this person is. Speaker verification considers the case where an identity claim is put forward as well as his voice. The system's task is then to verify if the claim put forward is correct.

There are two different types of speaker recognition systems. These are text-independent and text-dependent systems. The two different types have their advantages and disadvantages.

2.4.1 Text-independent

A text-independent system is able to accept variations in the spoken phrases. It is independent of the text spoken. Such a system can typically be used for providing verification or identification of a person by examining ongoing speech. An example can be a telephone service where a user calls in and provides an identity. While the user speaks with for example an operator,

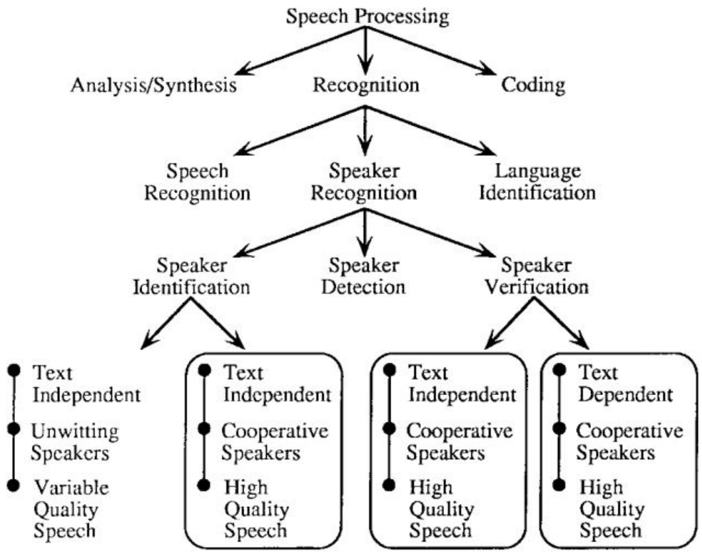

Figure 2.2: Different types of speech processing

the text-independent system can work in the background and verify if the user is who he claims to be.

The speech input in a text-independent system typically needs to be quite much longer than that of a system that is text-dependent. The error rates of such systems also tend to be a bit higher than for a text-dependent system.

In a text-independent system, the features used for comparison are typically found by averaging the features of the voice spoken, for the entire recording.

2.4.2 Text-dependent

In a text-dependent system the spoken phrase must be the same each time. The use of a text-dependent system therefore challenges a user to provide two types of information to be authenticated. In addition to being the correct speaker, the user must know the correct passphrase. A text-dependent system therefore provides two of three types of information described in Section 2.2, both information the person knows, and information the person is.

The features in a text-dependent system are typically different than in a text-independent system. In the text-dependent case we need to look at the features of the voice in all the different parts of the recording, to see if the correct passphrase is used. Therefore we will, in the text-dependent case, have a set of features of the voice, instead of just one average feature.

In the implemented system it has been chosen to use text-dependent technology. The main reason for this is due to the fact that a text-dependent system is able to provide a higher level of security than what a text-independent system can provide. Another important reason is that for such a system, the use of a short and equal phrase each time is more user friendly than making the user provide quite long samples of speech each time.

2.5 Android

The platform used on the smartphone is Android [8]. The platform is a software stack containing an operating system, middleware and key applications, and is specifically designed for use on mobile devices. As of March 2012, Android is the most popular smartphone OS in America, according to the market research firm Nielsen [9]. The OS holds a market share of 48.5%. The platform is used by several of the biggest mobile makers today, such as HTC and Samsung. Android is an open platform, which makes everyone able to build applications that can run on the system. Applications on Android are made using the Java programming language. The version used in this project is Android 4.0 (Ice Cream Sandwich).

2.5.1 Architecture

Figure 2.3 shows the major components in Android. The figure is taken from [8]. The system contains several core applications, which are shipped with the system. These are applications such as a phone application, web browser, e-mail client, and several others. The application framework is the framework used to create the applications. The system also contains several libraries which can be used. The system runs on a Linux Kernel.

Android provides a communication model, which makes it possible for applications from same and different vendors to interact. For example, an application that needs to show a map can interact with for example Google's map application. The different applications communicate through messages called intents. Intents are sent between components in applications. Android applications are built up by several such components. Intents can either be explicit or implicit. Explicit intents are intents that explicitly states

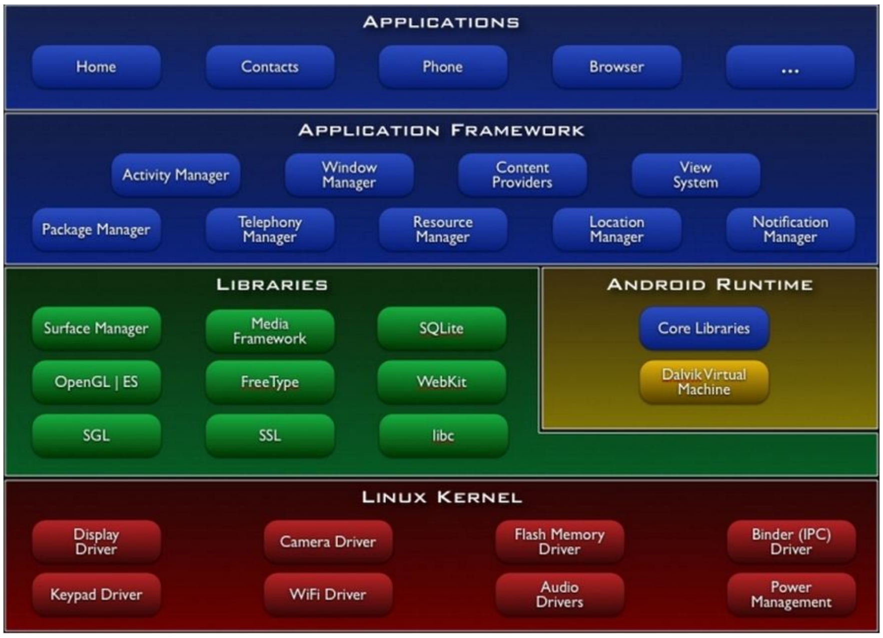

Figure 2.3: Major components of the Android Operating System

which components it wants to use. Implicit intents only states that it wants to use a component that has the specified functionality. If we use the map example, an explicit intent would be an intent that explicitly states that it wants to use Google's map component, while an implicit intent would state that it wants to use a component that has the map functionality, without specifying which one. Intents can also be used internally in an application, to invoke different components within the application.

Android has defined four component types:

Activity

Activities are the visual components in the system. These are the components the user interacts with, the user interfaces. An application typically contains several activities that together provide the entire user interface of the application.

Service

Service components run in the background, and have no direct interaction with the user. A service can for example process data which is then sent to an activity and shown to the user. As an example a service operation can be a file download which is done in the background while the user does something else on the application.

Content Provider

A content provider manages stored data, both for internal use within the application and for use between different applications. The data can be stored locally on the file system or other places accessible. Different applications can be able to fetch data from a content provider, depending on the restrictions set on the content provider.

Broadcast Receiver

A broadcast receiver manages broadcast messages sent throughout the system. It can also be used to manage the sending of intents directed to several receivers.

2.5.2 Android Security

Each Android application runs separately in its own secure sandbox. Having an own sandbox means that the applications have their own part of the system where they can run, and no other applications can access. In this case it means that the applications run their own virtual machine and Linux process, separating its running code from everybody else. All files the application contain are also by default not possible for others to access.

The open communication system in Android makes it susceptible to attacks by misusing the different messages sent throughout the system [10]. The different application components define permissions on which different types of intents it can receive. If these permissions are badly defined it can make the application susceptible to different attacks. For example, consider a component that handles sensitive information, which it passes on to certain other applications and components. If the permissions regarding which applications or components that are allowed to receive this information are defined badly, it could be possible for malicious applications to request and receive this information via intents. It is therefore important to define the permissions well, making sure only the correct components and applications have access.

2.6 System Performance

This section describes the different aspects of providing security in a biometric system. The use of biometrics creates several challenges regarding security. Some challenges are quite traditional in regards to providing security, while others are more unique for the use of biometrics.

2.6.1 Security Considerations

The ISO standard, ISO 24745, Biometric Information Protection [6], provides guidance for protecting biometric information. It provides three requirements regarding the security of the information. These are confidentiality, integrity and renewability/revocability.

Confidentiality

In this system, the ability to provide confidentiality means the ability to keep the data in the system secret. The biometric information should be kept confidential for multiple reasons. The InterNational Committee for Information Technology Standards provides three reasons in their report [11].

- Biometric characteristics are considered sensitive personal information, and should therefore not be shown in the clear.
- If the data is not kept confidential, it will provide the attacker with a digital copy of the biometric value. This can in some cases make attacks simpler to execute.
- If several applications use the same type of features in their authentication, it could be possible to acquire features from one application and then inject them into another application.

Integrity

Providing integrity of data means the ability to be certain of the source of the data, as well as being certain that the data has not been modified in any way. In this system this relates to the ability of being certain that the incoming voice sample is a live sample, and not a previously recorded sample. It is critical that we are able to check the integrity of an incoming voice recording. It should not be possible to use a recorded sample of the correct voice to get authenticated. The typical method used for checking the integrity of incoming biometric data is liveness detection. Liveness detection is a technique used for enabling detection of whether incoming biometric data comes from a living person or not. In the case of voice biometrics this would typically mean being able to separate a recorded voice from a live sample.

Renewability/Revocability

This requirement concerns the system's ability to renew and revoke data in the case of a database breach. If an attacker has gotten a hold of the reference of a user, it is desired that this reference is revoked and replaced with a new one. This can in many cases be hard to do when biometrics is used. For a text-independent system, renewability is a big challenge. The reference in such a system will consist of the features of the individual's voice, independent of any text. It is therefore very difficult to provide more than one feature set for each individual, making renewability very hard in such a system. However, in a text-dependent system, renewability is possible. Remember that in the text-dependent case we are dependent on the phrase spoken. Therefore it is possible to create several feature sets for an individual by making him enroll several sentences. Should a reference be disclosed, the system can replace the reference with a reference for another sentence.

Measurement Variations

One of the major problems in using voice, or any other type of biometrics as an authentication method, is the variations that arise between measurements of the biometric value. Different recordings of the same voice will typically have some small differences between them. This could occur because of several factors, such as unwanted noise, or the use of a different microphone. In a text-dependent system, differences in how the correct sentence or word is spoken can create variances. For example a vowel can be spoken longer than in the reference, there can be longer breaks between words in a sentence, and so on. These variations can create false acceptances and false rejections. A false acceptance can occur in two different cases, depending on whether the system uses identification or verification. If identification is used, a false acceptance occurs when a user is identified as a user other than himself. This means that in this case a user enrolled in the system can create a false acceptance. In the case of verification a false acceptance happens when an illegitimate user gets verified. A false rejection occurs when a legitimate user of the system is either not identified correctly, or not verified.

Because of these occurrences the system needs to set a decision threshold. This threshold defines how much an incoming template can vary from the reference. If the template has a difference from the reference within the threshold, it is accepted. If the distance is higher than the threshold, it is rejected. If an illegitimate user has a template that, when compared with the reference, has a distance within the threshold we have a false acceptance. If a legitimate user provides a template with a distance above the threshold we have a false rejection. The rates in which a false acceptance occurs is typically called False Acceptance Rate (FAR) or False Match Rate (FMR). The rate for a false rejection is often called False Rejection Rate (FRR) or False Non-Match Rate (FNMR).

Figure 2.4, from [12] shows two different distributions. The green line shows the distribution of comparing templates coming from the same user. The red line shows the distribution of templates from different users. We see that there is some overlapping between the two distributions. These overlaps create the false acceptances and false rejections. The defined threshold can be seen as the value T in the figure. The overlap that occurs to the left of T is the FRR, shown in the green field. The overlap on the right side is the FAR, and is shown in the red field.

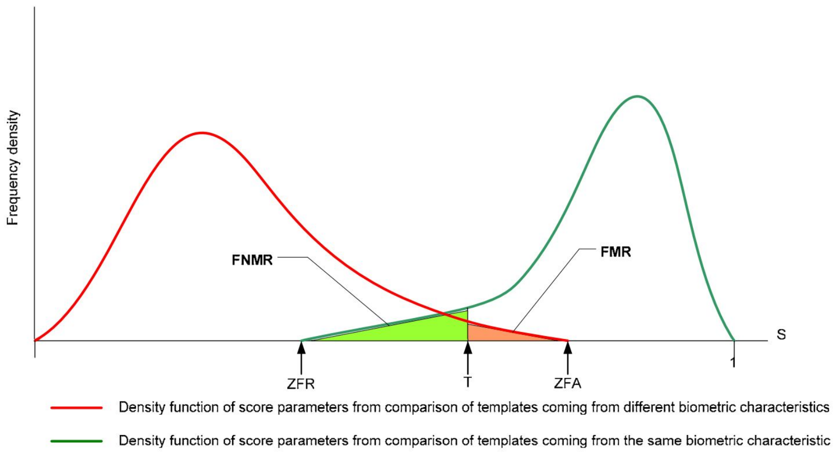

Figure 2.4: Typical distribution for score parameters

Varying the threshold will create changes in the rates of FAR and FRR. If the threshold is set quite low only small variations from the reference are allowed. This will make it harder for an illegitimate user to get accepted, but at the same time the legitimate user will be rejected more often. Therefore, in this case, the FAR will decrease while the FRR will increase. If the threshold is set too high you will get the opposite case with an increasing FAR and decreasing FRR. We therefore get a trade-off regarding where to set the threshold. Set it high and the correct user will get good usability by almost always be accepted, but at the same time the security is degraded since the FAR increases. If the threshold is set low the usability will be degraded since the correct user more often will be rejected, but the security of the system will be better. What the threshold should be set to therefore depends on what the specific system desires.

2.6.2 Privacy Considerations

Biometrics can in many cases be viewed as unique identifiers of an individual. This can be a very good attribute with regard to security, but creates problems regarding the privacy of the user. Biometric information is viewed as personal and sensitive information, and can give away important information about a person. This, along with the fact that biometric information in itself can identify a person, is why it is very important to address the privacy of the information stored in the system.

Biometric information can also contain some other information regarding the person. One example of this is health related information. The Fidis deliverable, Biometrics in Identity Management [13] has a list of such information that can be found in different biometric characteristics. On voice recognition it states that it can be possible to find information on diseases of the nervous system, such as Parkinson's disease and stroke.

The ISO standard, Biometric Information Protection [6] has listed three requirements that should be fulfilled in order to protect the privacy of the users. These are irreversibility, unlinkability and confidentiality.

Irreversibility

The irreversibility requirement states that it should not be possible to reverse the stored reference and find the original data. In this system this means that it should not be possible to derive the original recording of the voice by analyzing the stored reference.

Unlinkability

This requirement states that it should not be possible to link biometric references across different applications.

Confidentiality

The references should be kept confidential. It should not be possible for someone to get a hold of the reference in plain-text.

Chapter 3

System Architecture

This chapter will describe the architecture of the system built. The first sections describe the general architecture, while the last sections describe in detail the different algorithms used in the system.

3.1 Modular Audio Recognition Framework

The system makes use of the Modular Audio Recognition Framework [2], denoted as MARF. MARF is an open-source framework containing many algorithms that can be used for processing sound and speech. It also provides a good framework for adding new algorithms to be used.

There are several reasons why this framework is adapted in the system. First of all, it contains implementations of many of the algorithms needed to build a text-dependent speaker recognition system. The framework is also programmed using the Java programming language, which is also used for the implementation of this system.

The process in MARF was originally built for text-independent speaker recognition, but can be modified for use in a text-dependent system. The modifications we have done will be described later.

3.2 MARF Architecture

This section describes the authentication and enrolment processes in MARF.

3.2.1 Authentication

Figure 3.1 shows the general pipeline in MARF for the authentication process. The voice sample is first recorded and then sent as input to MARF. The sample is first preprocessed, and the features of the voice sample are then extracted. The features are then sent to the comparison stage for comparison with the stored reference features. The result calculated from the comparison is used in the decision process.

The framework has implementations of several algorithms for each of the stages described in the figure.

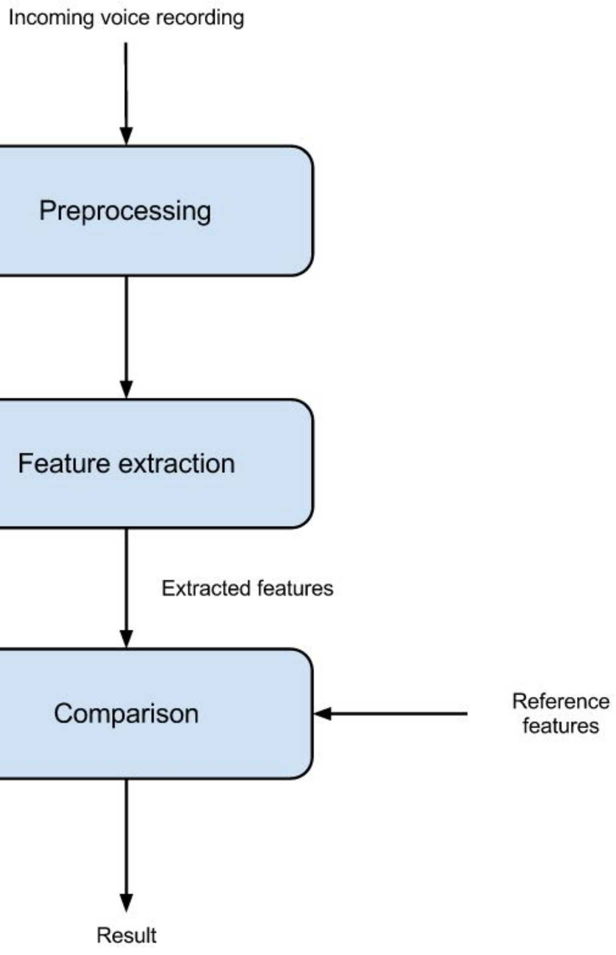

Figure 3.1: Core pipeline in MARF

3.2.2 Enrolment

The enrolment process in MARF uses the same pipeline as the authentication process, with the exception of the comparison stage. The purpose of the enrolment process is to create the reference that later will be used in the authentication process. It is similar to the enrolment process shown in Figure 2.1, in Chapter 2. The MARF enrolment process follows the pipeline in Figure 3.1 through the feature extraction stage. After this stage the calculated features are stored as the reference for the user.

MARF has a training feature which makes it possible to use more than one recording to create the reference. The training feature makes it possible to invoke the enrolment process more than once for each user. The first time training is called for a user, the process is equal to the enrolment process described above. After the first time, the reference will already have been created when the process is called. The difference will then be that the reference is updated instead of created. The reference is updated by finding the average features of the different samples used in the training process.

3.3 System Architecture

This section explains how MARF has been used in the implemented system. The authentication and enrolment processes are described.

3.3.1 Authentication

The implemented system uses the same stages as MARF in the authentication process. An explanation of how these stages are altered to provide a text-dependent system follows.

The preprocessing stage has not been altered in the implemented system. The preprocessing of a voice sample is in general no different in a text-dependent system from a text-independent system. This component consists of several algorithms that are implemented in MARF. These are normalization, endpointing and silence removal.

The feature extraction component makes use of some algorithms from MARF, but with some modifications. Since the extraction methods in MARF are implemented for use in a text-independent system, there is a need for some changes to make them work in a text-dependent system. The text-independent algorithms only output the average feature vector for the whole sample. In a text-dependent system we are in need of getting all the feature vectors from the sample. The algorithms are therefore changed to output all these feature vectors. The system uses two algorithms in the feature extraction stage, Hamming Window and Linear Predictive Coding (LPC). The Hamming Window algorithm is used to process data before the actual extraction and is therefore not modified from the MARF implementation. The LPC algorithm has been modified to output the entire set of feature vectors instead only the average vector.

The comparison stage needs more changes than the other two stages. Instead of comparing two single feature vectors, we are now comparing two sets of features. Therefore the algorithms implemented in MARF cannot be used for the main comparison. The Dynamic Time Warping (DTW) algorithm is implemented for this purpose. The Manhattan distance is used for local distance comparison within the DTW algorithm.

The pipeline and stages of the system is shown in Figure 3.2. In the next sections we will describe in detail the workings of the algorithms used in the different stages.

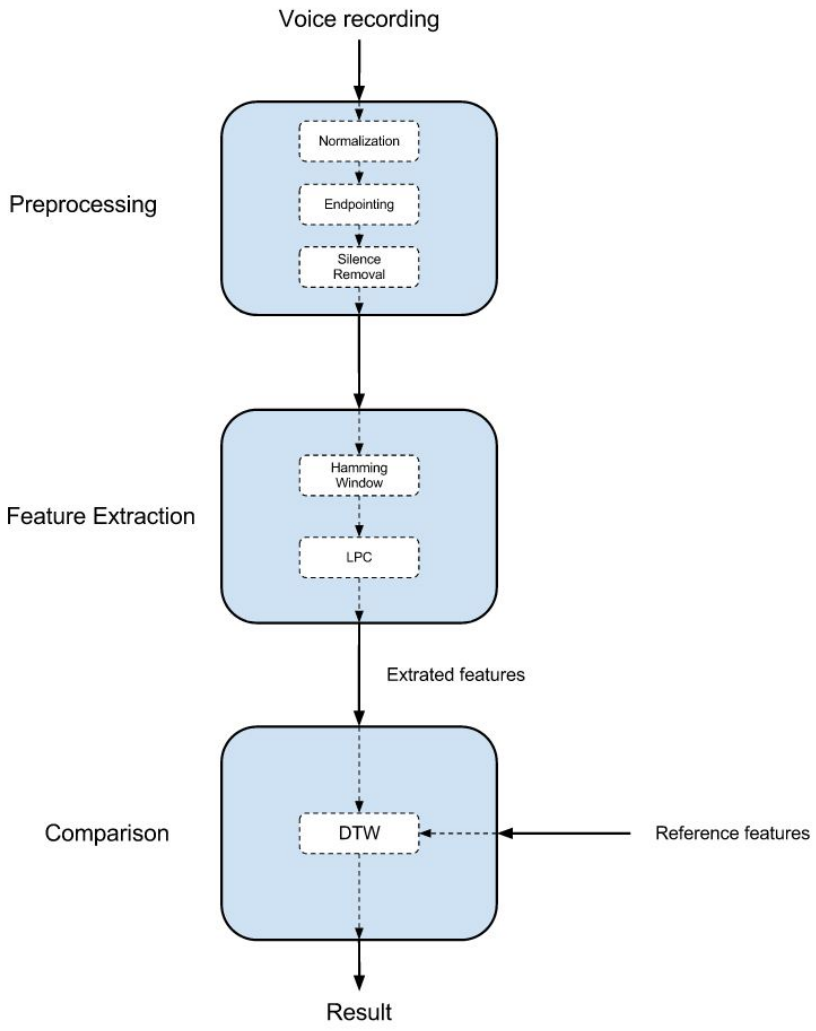

Figure 3.2: Core pipeline in the system

3.3.2 Enrolment

The enrolment process has been completely altered from the one found in MARF. The first steps in the process are still the same; the pipeline is followed until after the feature extraction stage. The difference lies in how the creation and updating of the reference is done. These tasks are performed by the training procedure implemented in the system. This training procedure is described in Chapter 4.

3.4 Preprocessing

The preprocessing stage is the first step in the processing of an incoming voice recording. There are three different methods that are implemented in this part of the system. These are normalization, silence removal and endpointing. The algorithms are described below. The implementation of each of these algorithms is taken from the MARF framework [2].

3.4.1 Normalization

The use of different microphones, recordings taken in different environments and other factors will create differences in voice recordings. One important factor is that the sound level of the voice typically will vary, creating different amplitudes for each sample. It is therefore important to normalize these differences in the amplitude. The method used for this normalization is to scale the amplitudes of the sample with regards to the maximum amplitude found in the sample. The implemented algorithm takes the maximum amplitude in the sample, and scales the sample by dividing each point with this value.

3.4.2 Silence Removal

Silence removal removes the parts of an incoming sample that are silent or almost silent. These parts are found by looking after places in the sample where the amplitudes are below a certain threshold. The threshold is set to 1% of the maximum amplitude. The values in the sample are denoted by points that are limited to (-1.0, 1.0), meaning the maximum amplitude is -1.0 or 1.0, making 1% equal to points within (-0.01, 0.01).

3.4.3 Endpointing

Endpointing is a technique that selects local maximum and minimum values in the sample, and removes the rest, thereby compressing the data. These local minimums and maximums are called end-points. The implementation of the algorithm works as follows:

The algorithm goes through each value in the sample, starting at the beginning. We denote the sample S. Each value in the sample is denoted S_i, where i is the index of the value. A value is chosen if one of the three conditions below is correct:

- $(S_{i-1} < S_i) \cap (S_{i+1} < S_i)$
- $(S_{i-1} > S_i) \cap (S_{i+1} > S_i)$

- $(S_{i-1} = S_i) \cup (S_{i+1} = S_i)$

The first statement is the condition for a local maximum, the value must be bigger than its neighbors. The second denotes a local minimum, meaning that the value is smaller than its neighbors. The last statement is added to enable continuous values to be chosen. In addition to these conditions, the values at the beginning and end of the sample are always chosen. Using this algorithm the sample is shortened, leaving us with the most important information.

3.5 Feature Extraction

After the preprocessing stage is finished, the features of the voice are extracted. The purpose is to extract the crucial information from the sample. This information makes us able to differentiate between different users. After this stage it is possible to discard the original sample. The extraction process will be slightly different depending on whether the system is based on text-dependent or text-independent authentication. How these two cases defer from each other will be explained later in the chapter. The following sections describe the different algorithms implemented and how they are used in the system.

3.5.1 Hamming Window

In a spectrum analysis of audio signals it is common to split the signal into shorter segments before any analysis is done. One particular advantage this gives is that it makes it possible to compare different templates in the time domain. To do this, one compares the features from the first segments in an incoming template with the features from the first segments of the reference, and so on, until the last segments. This makes it possible to perform a text-dependent analysis.

To create such segments, or windows, one typically uses a window function. The most simple window function creates a rectangular window. An example of a rectangular window with length $M = 21$ is shown in Figure 3.3, taken from [14]. A rectangular window sets all values outside the window to zero, and doesn't do any changes on the inside of the window. This will create a sudden drop to zero at the beginning and end of the window. This can often create some unwanted variations. To avoid this, one should use window functions that minimize the side lobes of the window [14]. The window function used in this system is Hamming Window. An example of a Hamming Window with length $M = 21$ is shown in Figure 3.4 [14].

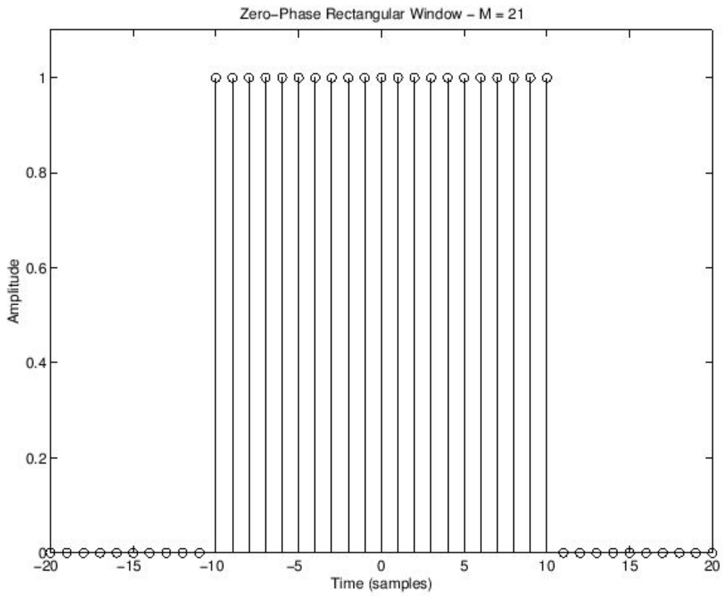

Figure 3.3: Rectangular window

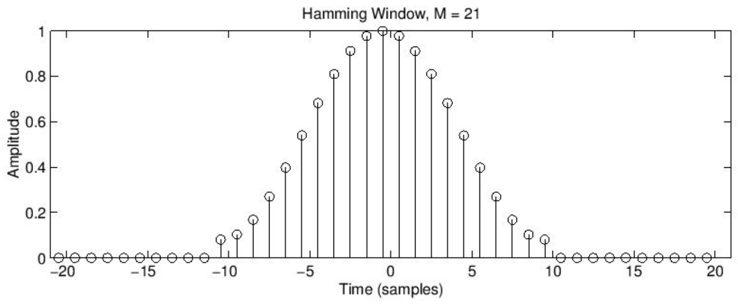

Figure 3.4: Hamming window

A Hamming Window is defined by the following function [14]:

$$x = 0.54 - 0.46 * cos(2 * \pi * n/(l-1)) \tag{3.1}$$

where x is the new sample amplitude, n is the window index and l denotes the length of the frame. It is also important that the different windows overlap somewhat so that each point is valued equally when the windows are summed up. In one window, the points on the edge will be degraded towards zero, and won't give a good representation of the value in itself. Therefore overlapping of windows is important.

3.5.2 Linear Predictive Coding

The feature extraction method used in this system is called Linear Predictive Coding (LPC). Using LPC we are able to retrieve the features from a speech sample. The features can be used for both text-independent and text-dependent authentication. LPC uses linear prediction to create these features from the sample. Linear prediction gives a representation of the voice with less data than the raw sample, and can for example be used for data compression [15]. These calculated representations can also be used as features of the voice.

Linear prediction takes as input a discrete time series signal denoted as s_n, where n is an integer varying with time. Linear prediction estimates future values of the signal s_n from linear combinations of past outputs and inputs from the signal. It can be described as [15]:

$$s_n = -\sum_{k=1}^{p} a_k s_{n-k} + G \sum_{i=0}^{q} b_i u_{n-i}, b_0 = 1 \tag{3.2}$$

Here s_{n-k} are the previously observed outputs from the system. a_k are the predictor coefficients. The value u_{n-i} denotes the past inputs to the system, and b_i are its corresponding predictor coefficients.

There are two cases of this model that are typically looked at:

- all-zero model, where $a_k = 0$
- all-pole model, where $b_i = 0$

The all-pole model is by far the most used of the two, and is also the one used in this system. The rest of the description focuses on this model.

In the all-pole model we have $b_i = 0$. The equation then becomes:

$$s_n = -\sum_{k=1}^{p} a_k s_{n-k} + G u_n \tag{3.3}$$

We see that the signal s_n is now estimated by a linear combination of its past values, and some input u_n. The number of poles used is denoted by p.

The values we are interested in calculating are the predictor coefficients a_k. The coefficients are used as the features of the sample. The system uses the least-square autocorrelation method to derive these coefficients:

It is assumed that the input u_n is unknown. So it is only possible to predict s_n approximately by the following equation:

$$\tilde{s}_n = -\sum_{k=1}^{p} a_k s_{n-k} \tag{3.4}$$

Since we are only able to approximate the value we will get an error between the approximated value and the correct one. We denote this error value to be e_n. The equation for the error will then be:

$$e_n = s_n - \tilde{s}_n = s_n + \sum_{k=1}^{p} a_k s_{n-k} \tag{3.5}$$

We are now able to calculate the coefficients a_k by minimizing this error. The details on the minimization procedure and the rest of the implementation of the algorithm can be found in the documentation on MARF [2].

Use in the system

As described in the previous section, we split the sample into smaller segments using a Hamming window. Each of these segments is then sent as input to the implemented LPC algorithm. The algorithm outputs a vector consisting of the predictor coefficients a_k for this segment. The vector will be of size p, corresponding to the amount of poles used.

The feature vectors are used differently in the implemented system than in MARF. In MARF the output from the LPC algorithm is one single feature vector calculated by finding the average predictor coefficients from all the segments. In the implemented system we are interested in each of the feature vectors from all the segments, instead of just the average feature vector.

Both this average feature vector and the entire set of feature vectors are sent as output from the extraction stage. The reason for this is that the average feature vector used in MARF can be of interest in the comparison stage, even though the main comparison is done on the entire feature set. Two average feature vectors can be compared using a method that can calculate the distance between two vectors. In this system the Manhattan distance is used for this purpose. The distance between two average feature vectors is, in this report, denoted as the average features distance. How such average feature vectors can be used for authentication in the system is described in later chapters.

3.6 Comparison

Comparison is the step that calculates the difference between the extracted features and the reference, and determines a comparison score. The comparison score is used for determining whether the user is authenticated or not. The stage compares the extracted features derived from the previous stages with the reference features. This process can be different depending on whether identification or verification is used. In the identification case, comparison must be done against the references of all the enrolled users of the system. In the case of verification one only needs to compare against the claimed identity reference.

As mentioned, the comparison done in a text-dependent system is quite different than in a text-independent system. In the text-independent case, the output from the feature extraction step is a single average feature vector. To compare two such vectors there is only a need for an algorithm able to calculate the difference between two vectors. The Manhattan distance is used for this purpose. In this system, the extracted features are sequences of feature vectors from the entire voice sample. Here, an algorithm which is able to compare time-series of data is needed. The Dynamic Time Warping (DTW) algorithm is implemented for this case. This algorithm is explained next.

3.6.1 Dynamic Time Warping

Dynamic Time Warping is an algorithm that compares two vectors that may vary in time and speed. The algorithm finds an optimal path between the two vectors. The DTW algorithm is extensively used within biometrics, examples are speech recognition [16], fingerprint verification [17] and gait recognition [18]. The algorithm is used in this system because of its ability to find similarities between samples, even if the similarities are out of phase in the time domain. Most of the time there will be some shifts and variations between different utterances of a word or a sentence. For example, a vowel can be pronounced a little longer, or there can be a shorter pause between two words in a sentence. To be able to compare such utterances with each other it is very important to be able to find and correct such differences. This can be achieved by using the DTW algorithm. Next it is described how the algorithm works and how it can be used effectively in this system.

Algorithm

The algorithm takes as input two vectors $a = \{a_0, ..., a_n\}$ and $b = \{b_0, ..., b_m\}$. Its goal is to calculate an optimal path between the two vectors. First the algorithm calculates a local distance matrix, $d_{n\times m}$, consisting of the pairwise distances between all the points in the vectors. As an example, the distances $\{(a_0, b_0), (a_0, b_1), ..., (a_0, b_m)\}$ is calculated for the first point in a. The same is done for all the other points $\{a_1, ..., a_n\}$. In this system each of these points is in itself a feature vector, consisting of several points. Therefore, to be able to calculate the distance (a_n, b_m) one needs a local distance calculation method able to calculate the difference between two vectors. In this system the Manhattan distance is implemented for this purpose. The algorithm for the Manhattan distance is described later in this chapter.

After $d_{n\times m}$ is found, the algorithm calculates a global distance matrix, denoted $D_{n\times m}$. Each point in this matrix denotes the minimum distance to the first point, (0, 0). $D(n)(m)$ denotes the minimum distance from the last point, (n, m), to the first point, (0, 0). $D(n)(m)$ is therefore the minimum distance between all the points in the two vectors. The matrix is calculated by the following three equations [19]:

$$D(i,0) = \sum_{1}^{i} d(i,0) \tag{3.6}$$

$$D(j,0) = \sum_{1}^{j} d(j,0) \tag{3.7}$$

$$D(i,j) = min \begin{cases} D(i,j-1) + d(i,j) \\ D(i-1,j-1) + 2d(i,j) \\ D(i-1,j) + d(i,j) \end{cases} \tag{3.8}$$

Since we know each point in D corresponds to the minimum distance from the point to $(0,0)$, we are able to calculate the optimal path from the matrix. We start at the last point (n,m) and choose the neighbor with the minimum distance, that is:

$$min\,\{D(n-1)(m), D(n)(m-1), D(n-1)(m-1)\} \tag{3.9}$$

We then repeat the process for this chosen point, choosing its neighbor with the minimum distance. This is repeated until we reach (0, 0). This is an example of a greedy algorithm, where we always choose the locally optimal solution to find the optimal global solution.

We now have calculated an optimal path between the two compared sets of features. This optimal path is called the warping path. Figure 3.5 shows an example of such a warping path. In the figure, the extracted feature vectors are shown along the horizontal axis, and the reference features along the vertical axis. The incoming features are in this case longer than the reference features, meaning that there are more steps in the direction of the incoming features in the path.

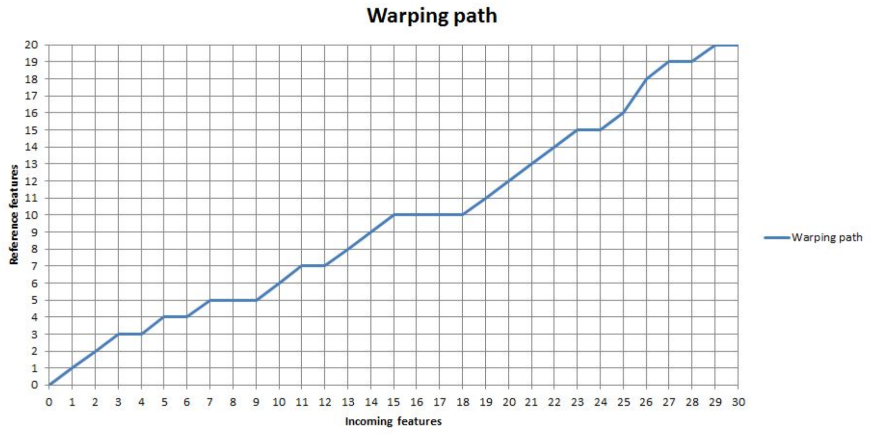

Figure 3.5: Example warping path calculated by DTW

Use in the system

There are two different distances outputted from the DTW algorithm that are of interest in regards to the authentication process. These are the accumulated distance and the average step distance.

Accumulated Distance The accumulated distance is equivalent to the minimum distance $D(n, m)$. This distance is the entire distance covered by the warping path. If we use Figure 3.5 as an example, the accumulated distance will be the sum of the cost of each step taken in the shown warping path. The cost of each step denotes the distance calculated between the feature vectors in each step. One important thing to notice regarding the use of this distance is its sensitivity to the length of the features compared. The distance summarizes all the steps, meaning that on average the distance will increase when the amount of steps increases, and decrease when there are fewer steps. This means that a short utterance of the correct sentence, creating a small set of incoming features, on average will get a lower score than that of a longer utterance of the same sentence.

Average Step Distance The average step distance is also calculated from the warping path. The distance is the average cost of each step in the warping path. It is calculated by taking the accumulated distance and dividing it by the amount of steps taken in the warping path. This distance is less susceptible to differences in the length of the features compared, since it is not dependent on the amount of steps taken.

How these values are used in the authentication process will be described in later chapters.

The implementation of DTW in the system is a modified version of the open-source JAVA version found on Koders.com [20]. The algorithm is modified so that the input to the algorithm is two-dimensional, making it able to process a vector containing feature vectors, instead of only one vector. The calculation method used to calculate the distances in the $d_{n \times m}$ matrix is modified to use the Manhattan distance. This distance classifier is described next.

3.6.2 Manhattan Distance

Manhattan distance is, as mentioned, used as the local distance classifier between two feature vectors from the set of feature vectors used in the DTW algorithm. Manhattan distance is also known as City block distance. The Manhattan distance is calculated by taking the distance between each corresponding points in the vectors and summarizing these. Equation 3.10 shows the calculation of the Manhattan distance between two vectors.

$$d(a,b) = \sum_{i=1}^{n} (|a_i - b_i|) \tag{3.10}$$

where a and b denote the vectors compared, and n is the dimension of the vectors.

The implementation of this algorithm is taken from the MARF framework [2].

Chapter 4

Technical Procedure

This chapter describes the different implementations done in the system.

4.1 MARF implementation

MARF is a framework built for use on computers. Therefore it needs some modifications to be able to run on the Android platform. In particular, MARF uses the Java Sound API [21], which is not supported by Android. The API is a low-level API providing functionality for the handling of audio files. More specifically, MARF uses the classes AudioFormat and AudioInputStream from the API. The AudioFormat class is used for specifying the format of different audio files. The AudioInputStream class provides functionality for reading byte-arrays. AudioFormat is used because MARF provides support for several file formats. In this system all recordings are done in the same format. Therefore we are not in need of providing functionality for several formats. The functionality provided by AudioFormat was therefore removed. The functionality provided by AudioInputStream is still needed in the system, since the ability to read the audio files is critical. Therefore this functionality needed to be replaced with similar functionality that is supported by Android. The Apache Commons IO library [22] was used for this purpose.

MARF contains functionality for storing the references for each user. This functionality is used in the system with some modifications. The storage functionality has been altered to enable storage of files on the Android platform. Remember that each application in Android runs in its own sandbox. This means that each application has one specific location where it is able to store files. The storage functionality is therefore modified so that all files will be stored in this location.

4.2 Training procedure

A training procedure for the system was implemented. This is the enrolment procedure in the system. The procedure makes it possible to calculate an average reference from several samples of the correct utterance. The next sections describe the core algorithm implemented as well as the different processes the algorithm can be used in.

4.2.1 Algorithm

The goal of the training algorithm is to be able to compute an average reference from several utterances of the correct passphrase. Each of these utterances will create a set of features. The task of the training algorithm will be to align these different sets, and compute an average set of feature vectors from them.

The utterances of the correct passphrase are typically a bit different from each other. Examples of differences are time shifts, differences in the length of the utterance, words spoken more quickly or longer, and several others. These differences need to be found and corrected if the reference should work properly. The training algorithm makes use of the DTW algorithm to find and correct these differences.

The training algorithm takes as input the extracted features of a voice recording. The algorithm behaves differently depending on whether this recording is the first to be trained on or not. The first recording trained on creates the reference for the user. All other training samples only update this reference.

The extracted features of the first recording are stored as the reference for the individual. When the next training sample comes as input, the process will be different. As explained, the features from this sample needs to be aligned to the stored reference features. The incoming features and the reference features are therefore sent as input to the DTW algorithm. The output from the DTW algorithm of interest in this case, is the warping path. The path describes how the incoming features are aligned to the reference features. An example of such a path is shown in Table 4.1.

In this case, both the reference and the incoming features consist of a set of 6 feature vectors. The numbers in the table describe the index of the features in the time domain, where 0 is the first vector and 5 is the last vector. The warping path consists of 7 steps. The rows in the table show

Reference features	Incoming features
0	0
1	1
1	2
2	3
3	4
4	4
5	5

Table 4.1: Example of a warping path

each step. The first step is (0, 0) and the last step is (5, 5). We now choose the incoming features that correspond to the indexes of the reference. For index 0, we choose the vector from the incoming features with index 0. For index 1 there are two feature vectors that correspond. In such cases we calculate the average of these vectors. We therefore calculate the average of index 1 and 2 from the incoming features. This process is performed for all the indexes of the reference. When this is completed, we have a set of aligned features equal to the length of the reference. The aligned features are shown in Table 4.2.

Reference features	Aligned incoming features
0	0
1	1, 2
2	3
3	4
4	4
5	5

Table 4.2: Aligned features

We are now able to update the reference with these aligned features. The reference will consist of the average feature vector for each of the indexes. The update will be to compute the new average when these new aligned features are added to the set of considered samples.

The ability to train on several samples will make the system's ability to recognize the correct individual better and more stable. If only one sample is used as the reference, the risk is that the sample could be quite far from the average value. This case will create larger differences between correct samples and the reference. As an example, we consider a test done on the system. The test is done using two persons, and shows the system's ability to distinguish them from each other. Table 4.3 shows the results from the first test, where only one sample from each person is used as the reference. The left columns show the average distances from Person 1's reference for samples from Person 1 and Person 2. The right columns show the same for Person 2. The distance shown is the accumulated distance.

Person 1		Person 2	
Person 1	Person 2	Person 1	Person 2
35,6	49,62	46,33	39

Table 4.3: Average accumulated distance from reference

Table 4.4 shows the results from the second test, where the reference for each user was created by sending several samples from each person as input in the training algorithm.

Person 1		Person 2	
Person 1	Person 2	Person 1	Person 2
25,43	41,56	42	29,25

Table 4.4: Average accumulated distance from reference with new training procedure

We see that the average distance from the reference goes drastically down when training is used.

4.2.2 Training Processes

There have been implemented several training processes that utilizes the described training algorithm. The first training process that was implemented is an iterative process, where the training algorithm is called each time the user records a new training sample.

The use of this training process creates some problems because of how the length of the reference is set. The reference length is entirely based on the length of the first sample sent as input to the training algorithm. This creates problems when the length of this sample is much longer or shorter than the average length of a correct utterance. When the described training process is used, the length of the reference will be determined by the first recording done by the user. As an example of the problem this creates, we consider a test done on the system where this training process was used. The test consisted of two persons each recording six samples of the same sentence. Both persons reference was created by using all their samples. Each sample was then compared against both references. Tables 4.5 and 4.6 show the results from this test. Table 4.5 shows the results when the accumulated distance is used, while Table 4.6 shows the same for the average step distance. The left columns show the length of each of the samples recorded by the user, and their distance from the reference of both Person 1 and Person 2. The right columns show the same for Person 2. The last row in both tables shows the average values for each of the columns.

Person 1			Person 2		
Length	Person 1	Person 2	Length	Person 1	Person 2
23	23	33	33	41	33
28	26	38	23	32	31
27	30	42	22	31	29
26	24	36	21	26	28
25	29	40	24	30	29
29	30	41	25	29	27
26.33	**27**	**38.33**	**24.67**	**31.50**	**29.50**

Table 4.5: Accumulated distances from reference

Person 1			Person 2		
Length	Person 1	Person 2	Length	Person 1	Person 2
23	0.96	0.94	33	1.22	0.97
28	0.87	1.03	23	1.3	0.92
27	1.09	1.23	22	1.2	0.86
26	0.86	0.99	21	1.1	0.84
25	1.07	1.18	24	1.17	0.84
29	0.98	1.21	25	1.08	0.77
26.33	**0.97**	**1.09**	**24.67**	**1.18**	**0.86**

Table 4.6: Average step distances from reference

The reference has the length of the features from the first sample. We see that the first samples of both persons have a length quite different from the others, thereby creating some unwanted effects in the results. We see that the reference for Person 2 is longer than his other samples. This creates on average a quite high accumulated distance between his other samples and the reference. For Person 1, the reference is shorter than the other samples, making the average accumulated distance also shorter. The problem with this short reference length is that it creates a low average accumulated distance for Person 2's samples compared against Person 1's reference.

As described in Chapter 3, the average step distance will not vary as much with the sample length as what the accumulated distance does. This can be seen in the results, where the difference between the average distance of Person 1 and 2's samples when compared to Person 1's reference has a greater distinction when the average step distance is used.

To counter this length problem, we implement a new process which takes as input a set of samples to be trained on, instead of only one at a time. This makes it possible to select a more appropriate length for the reference. The new process first extracts the features for each of the samples. It then finds the sample with the median length of features. This length is chosen as the length of the reference. Remember that the length of the reference is decided by the length of the sample first processed by the training algorithm. Therefore the sample with the median length of features is sent as input to the training algorithm first, thereby creating the reference. The length of the reference is now set. The rest of the training samples are then sent as input to the training algorithm, one by one.

A test using this new training process was also performed. The samples used in this test are the same as in the previous one. The test results are shown in Table 4.7 and Table 4.8.

We see that by using the new training method, the average accumulated distance between Person 2's samples and his reference has decreased significantly. We also see that Person 2's distances from the reference of Person 1 now are bigger. The accumulated distances between the reference and Person 1's samples have become a bit larger. The reason for this is the increase in the reference length, creating a longer path to traverse. We see that this only occurs with the accumulated distance, since the average step distance on average has decreased from 0.97 to 0.92.

Another advantage achieved by using this training process is that it enables limiting of samples that are possible to train on. A limit can be set on

Person 1			Person 2		
Length	Person 1	Person 2	Length	Person 1	Person 2
23	31	34	33	41	35
28	29	38	23	40	24
27	28	32	22	36	25
26	25	37	21	32	22
25	28	39	24	37	21
29	28	41	25	35	21
26.33	**28.1**	**36.83**	**24.67**	**36.83**	**24.67**

Table 4.7: Accumulated distances from reference

Person 1			Person 2		
Length	Person 1	Person 2	Length	Person 1	Person 2
23	0.96	1.09	33	1.22	1.02
28	1.02	1.33	23	1.38	0.92
27	0.89	1.06	22	1.29	0.90
26	0.86	1.30	21	1.16	0.91
25	0.91	1.31	24	1.24	0.86
29	0.92	1.37	25	1.19	0.82
26.33	**0.92**	**1.24**	**24.67**	**1.24**	**0.90**

Table 4.8: Average step distances from reference

how much shorter or longer a sample can be compared to the reference. If a training sample does not meet these thresholds, it is not used for training. This makes us able to avoid using bad samples that can corrupt the average values in the reference.

This new training process is the one used to create references in all the different tests described in the next chapters.

4.3 Personal thresholds

The implementation of the training process made it possible to introduce personal thresholds. A threshold is, as explained in Chapter 2, the maximum allowed distance between the reference and an incoming sample. If the distance is above this threshold, the sample is rejected. Such thresholds are typically set to the same value for all users in the system. A personal threshold is a threshold set only for one specific person. The use of a default threshold can create varying performance in the system for different users. If a user has a tendency to have big variations in the way he speaks the passphrase, it is possible that his average distance score could be higher than usual. With a higher average distance score, the user would experience getting distance scores above the threshold more frequently than other users, creating an increased FRR. This would degrade the system's usability for this user. In such cases it would be desirable to introduce a personal threshold for the user. The threshold would be set higher than the default threshold, degrading the user's FRR to a normal level, making the system usable also for this person.

The challenge with using such personal thresholds is that they need to be set during enrolment. The value of the threshold should optimally be found by looking at the average distance score for samples compared to the user's own reference. The average distance score would be calculated by looking at the different scores the user achieves during the authentication phase. Since the threshold needs to be set before this phase, this is not possible.

The only samples possible to use, are the ones collected in the training process. The problem with using these samples is that their score will be different than an average distance score from "fresh" samples, since the samples used in the training process are included in the reference, creating on average a lower score than samples not trained on. These samples should still be able to give a good picture of the average distance of new samples.

The training process was altered so that after the reference is created and updated with each of the samples, the same samples are compared with the newly created reference, giving a distance score. The average distance score is then calculated. This score is used to determine what the threshold should be set to for this user.

The analysis of how the scores from the training samples can be used for determining thresholds is presented in Chapter 6.

4.4 Testing framework

A testing framework has been implemented for use in the system. This framework has been used in all the different tests performed on the system. The original application is built for running on only one voice recording at a time. In the application no voice samples should be stored beyond the authentication process. This becomes a problem during testing of the system. If it is only possible to run each sample one time, it will be very time consuming to perform a thorough test. Therefore we are in need of a test framework able to store voice samples for use in more than one test. In addition to this, it is beneficial to run all tests on the same set of samples, instead of having to change the set of samples for each test. This makes it easier to compare the results from the different tests.

In the testing framework, the recording of samples is separated from the authentication and training process. This is done to make the recording process easier for persons using the test framework.

Figure 4.1 shows the user interface for voice recording in the testing framework. It consists of two sliders and two buttons. The buttons are used for starting and stopping the recording. The first slider represents the ID of the person recording the voice. This ID is used to make it possible to know which person the recording originates from. This information is important to have later in the testing process. For each different person the system is tested on, a new ID is chosen. The second slider represents the amount of samples recorded for the specific ID. The value is incremented automatically after each recording done.

Figure 4.2 shows the interface for the authentication and training process in the testing framework. The interface contains three sliders and two buttons. The two top sliders represent the same values as the two shown in the recording interface. These two sliders make it possible to navigate through the different ID's and recordings. The third slider makes it possible to select different sets of references. It is important to be able to have several such sets of references. As an example of why, consider the testing of the different training processes. Each process will create its own set of references. When all the stored samples will be tested, they can be tested against the references from the different training processes. In this way we are able to compare the different procedures. The use of different sets of references also makes us able to vary the algorithms used in the different processes and compare the different methods against each other.

The train button trains on either specific samples or sets of samples, depending on which training process that is used. The authenticate button runs the specified sample through the authentication process. It is then compared with the reference stored in the specified reference set.

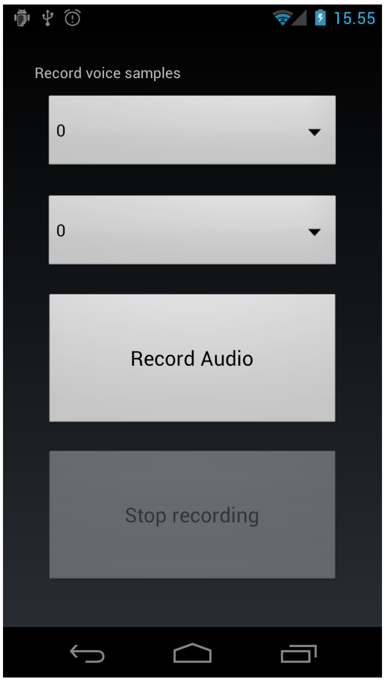

Figure 4.1: Interface of recording in the test framework

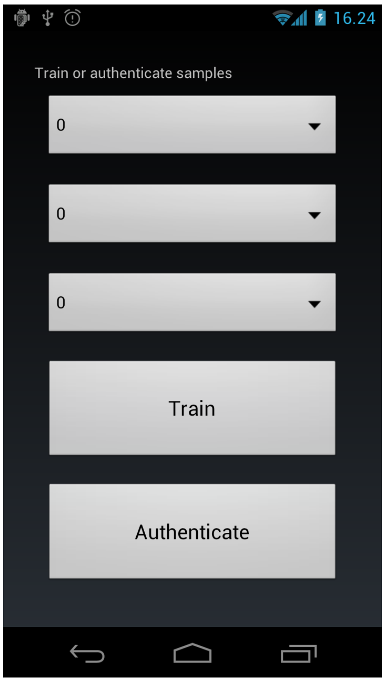

Figure 4.2: Interface of the authentication and training in the test framework

Chapter 5

Experimental Procedure

This chapter describes the different experimenting and testing done on the system.

5.1 Testing of different utterances

Several different utterances have been tested for their performance. The goal for this part of the testing is to see if there are some distinct differences in performance between different types of phrases.

5.1.1 Numbers

The use of numbers was tested. The idea is to have the user enroll each digit from 0 to 9. During authentication the user would be prompted to speak five of these numbers chosen at random. By using this challenge-response technique it would be harder for an imposter to use recordings of the correct voice. The imposter would need to have recordings of each of the different numbers.

It was observed that there typically were big variations in the distance score for correct samples when numbers were used. These big variations made it difficult to distinguish between the numbers. The reason for the big variations was most likely caused by the very short length of the utterances, making it easier to get big variations in the recordings. Numbers was therefore not chosen for the implementation.

5.1.2 Words

The use of single words was also tested. The idea is the same as for the numbers. The user would train on a set of different words, and at the authentication phase would be prompted to speak some of them.

Tests showed that the use of single words had the same problem as numbers. The utterance is too short to be able to avoid big variations, and therefore it is difficult to achieve consistent results.

5.1.3 Sentences

Different sentences of varying length were tested and compared against the performance of single words and numbers. The testing showed that the distance score for correct samples had less variation when sentences were used. The tests also showed that the system was able to better distinguish between different sentences than what was the case between numbers and words.

After testing several different types of utterances, it was found that the best performance was achieved using a sentence. In the main testing, the correct passphrace is therefore a sentence.

5.2 Testing of different noises

During the experimentation, several tests with recordings of different noises were performed. These noises would vary in length and content. Some were monotone, while others had much variation. In some cases the input was only silence in different lengths. These tests were performed to measure the system's ability to handle different types of input. The different samples of noise were tested against several references of legitimate users. The results of these tests was then compared against the typical results of legitimate samples.

5.2.1 Very short samples

Very short samples often created quite small distance scores. In some cases the calculated distance was quite close to distances obtained by legitimate users, especially when the accumulated distance was used. This problem mainly occurs because of the accumulated distance's sensitivity to variations in the length of the incoming sample, as well as the length of the compared reference. Recall that the accumulated distance is calculated by adding up the distances from all the different steps taken in the warping path. The distance therefore varies with the amount of steps in the warping path. This means that a long warping path typically will create a longer accumulated distance. The length of the warping path is dependent on the length of the incoming features and the reference. This means that if the incoming features are very short, the warping path will also often be of a small size. This could in many cases create a small accumulated distance, even in the case where the incoming features bear no resemblance to the reference.

5.2.2 Very long samples

Very long samples had a good probability of creating quite good scores when the average step distance was used as the classifying distance. Figure 5.1 shows such a case from the testing. The incoming sample is three times as big as the compared reference. The sample consists of a monotone sound and bears no resemblance to the reference. The reference has a length of 100 feature vectors, and the incoming sample consist of 300 feature vectors.

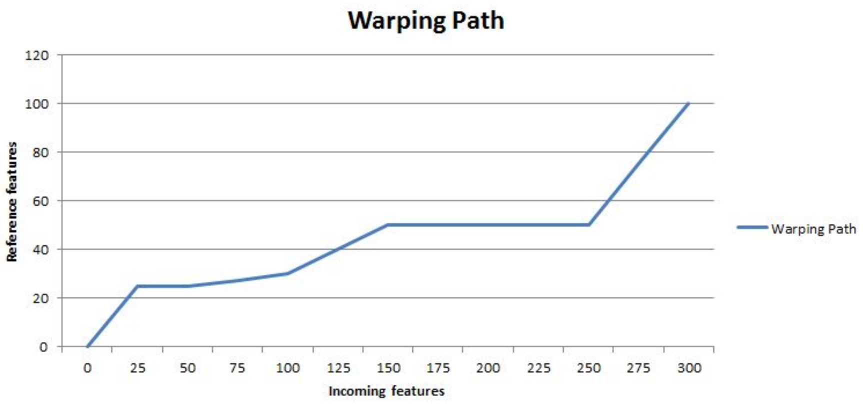

Figure 5.1: The DTW warping path between two sets of features

Since the incoming sample is quite monotone it will typically consist of feature vectors very similar to each other. Since the sample is much longer than the reference, there will be a lot of steps in the warping path that only goes in the direction of the incoming features, as seen in the figure. We see, for example, that the path follows the same reference feature between (50, 150) to (50, 250). In many cases the DTW algorithm will find a feature vector in the reference that aligns well with the vectors in the incoming reference, and follow this vector for many steps. A relatively big part of the steps taken in the path will therefore be between this reference vector and similar ones from the incoming features. Because of this the average step distance will end up quite small, even though the total accumulated distance will be very big.

The most extreme cases of this problem occurred when the incoming sample consisted of a monotone sound, as described above. Other long samples more varying in sound was also able to create quite good scores in many cases, so the problem was not only limited to samples with a monotone sound.

5.2.3 Countermeasures

The simplest solution to avoid such problems is to limit the possible length of an incoming sample. These problems only occur when the length of the incoming sample is very disproportionate to the reference length, as explained above. Therefore limitation of the length of an incoming sample is needed. The limit is specified with regard to how long or short a sample can be in comparison to the reference. The challenge when introducing such limitations is to avoid rejecting legitimate samples. Therefore the limits need to allow natural variations in the length or the correct uttered sentence. It was found that the optimal limits with regard to allowing such natural variations, was to let an incoming sample be able to vary with 40% from the reference. This means that an incoming sample can be 40% shorter or larger than the reference.

In all other cases than the two described above, samples containing silence or noise were not able to achieve scores close to the ones obtained by legitimate samples.

5.3 Comparing different sentences

Several tests have been performed to see how well the system is able to distinguish between different sentences. Since this system is built as a text-dependent system it should not be possible for a user to be accepted without speaking the correct phrase. Therefore it is very important that the system is able to separate between the correct sentence and all other phrases.

The tests were performed by making several users record samples of a number of sentences. These sentences were typically quite similar in type and length. This was done to be able to measure the system's performance in the extreme case where all sentences are very similar.

The results showed that the system performs quite well in separating sentences. There were few cases where speaking the wrong sentence would get a user accepted. The tests showed that the typical difference between different sentences spoken by the same user was quite equal to the typical distance between two different users speaking the same sentence.

5.4 Main test

The main test of the system was performed by testing on 16 different persons. Of the 16, 13 were men and 3 were women. All the recordings and calculations were done on the same smartphone. All the recordings also used the same microphone to avoid creating separation in samples because of varying input sources.

The test considers the case where everybody speaks the same sentence. This is in general a worse situation than the case where the attacker does not have the knowledge of the right sentence. The goal of the test is to calculate the rate in which the correct person is verified and the rate in which an illegitimate person is verified.

5.4.1 Procedure

Each person in the test recorded 15 samples of the same sentence. A reference was created for every person. The first ten samples recorded were used for the training procedure, the last five samples were used for the actual testing. The reason for only using the last five samples for testing is that they have not been used in the training procedure and therefore can be considered as totally independent from the reference.

5.4.2 Authentication

The goal of this test is to calculate the system's performance in regards to authentication. The two important rates to be considered in this case are the FAR and FRR. These two rates show the system's ability to reject malicious users as well as its ability to accept legitimate users. The FAR rate has its worst-case when all incoming samples uses the correct sentence. This is the case considered in this test. For each reference, the five samples coming from the correct person are considered as legitimate samples, while the 75 other samples coming from all the other persons are considered illegitimate samples.

We are interested in calculating what the typical distance score is for a person's sample when compared to his own reference. This is done by running each of a person's five samples against his own reference and calculating the score. By doing so for each person, we are able to calculate the distribution of this score.

It is also interesting to look into the typical distance score for illegitimate samples. This can be compared to the typical score for correct samples and give a good view of the difference between the two. For each reference, the 75 samples not coming from the correct person is run against the reference, creating a score. This makes it possible to calculate the distribution of this score as well.

The two distributions described above give a good measure of what the FAR and FRR will be in the system. As shown in Figure 2.4, in Chapter 2, the overlap between these two distributions create these two rates. The distributions make us able to calculate the optimal threshold that minimizes the FAR and FRR.

There are three differently calculated distances that are of interest in the system. The first two are the distances output from the DTW algorithm, the average step distance and the accumulated distance. The last distance of interest is the distance calculated when comparing the average feature vectors output from the LPC algorithm, as described in Chapter 3. This distance will be denoted as the average features distance.

For each of these distances, the distributions described above are calculated. This makes it possible to find the FAR and FRR for all three. Using this information, the distances are compared against each other with regard to performance analysis.

To improve the FAR and FRR, different combinations of the three distances are tested as well. The tests are performed by setting a threshold for each of the distances and then testing all correct and wrong samples against these thresholds. Several combinations are tested to find the optimal trade-off of the FAR and FRR.

The use of personal thresholds, as explained in Chapter 4, is also tested. First the average distance score for both samples used and not used in the training procedure is calculated. These results are then used to determine how the thresholds should be set. The use of these personal thresholds is then tested.

5.4.3 Identification

For testing the system's ability to identify the correct person, the following procedure was used. The five testing samples of each person are sent as input to the system. The system then calculates a result for this sample against each of the 16 references. The identification rate is then calculated by looking at how many of the samples that are correctly identified.

Chapter 6

Performance Evaluation

This chapter will first present the results obtained when performing the main test described in Chapter 5. These results are then discussed with regards to the performance of the system.

6.1 Results

6.1.1 Authentication

In this section show the distributions for the average step distance, accumulated distance and the average features distance. Note that in Figure 6.1 and Figure 6.2 the red lines for both the average step distance and accumulated distance do not add up to 1. This is because in around 25% of the comparison cases, samples are not within the limit for the size of incoming samples described in Chapter 5. In these cases, a comparison score is never calculated. This limit was not used when calculating the average features distance.

Average step distance

Figure 6.1 shows two distributions. The blue line shows the distribution of the average step distance when a user's test samples are compared against his own reference. The red line shows the distribution of the same distance when the user's test samples are compared with all the other users' references.

We see that samples compared to the correct reference on average create a shorter distance. Samples have a distance within 0.9 and 1.4 from the correct references, while the distance to the wrong references vary between 1 and 1.75. There is a significant overlap between the two distributions.

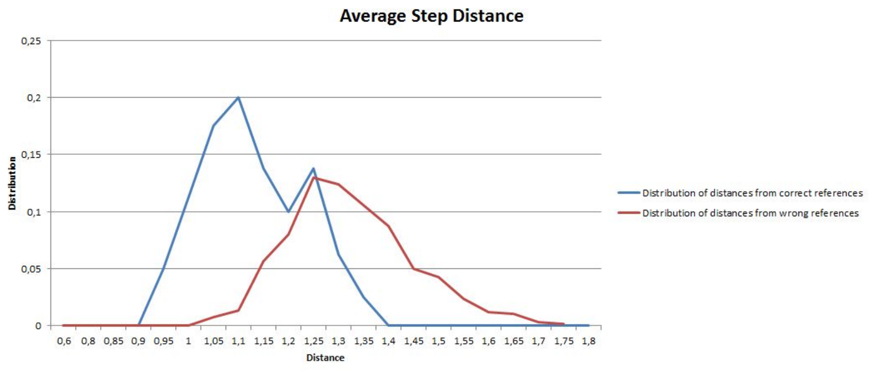

Figure 6.1: Distributions for the average step distance

This overlap creates the false acceptance and false rejection rates. These rates will depend on how high or low we set the threshold for the distance. If the threshold is set to 1.4 or bigger it will create a FRR on around 0%, but at the same time the FAR will be very high, around 60%. Different thresholds are therefore tested in order to find the optimal choice. Table 6.1 shows the FAR and FRR for differently selected thresholds.

Threshold	False acceptance rate	False rejection rate
1.15	7%	32%
1.20	15%	22%
1.25	28%	8%
1.30	41%	2%

Table 6.1: FAR and FRR for average step distance

We see that the optimal threshold for the average step distance is somewhere between 1.20 and 1.25.

Accumulated Distance

Figure 6.2 shows the same distributions when using the accumulated distance instead of the average step distance.

We see that the distances achieved when comparing samples against the correct reference are on average shorter than the ones for the accumulated distance. The distributions seem to have a bigger variation, at least for samples compared with wrong references. Samples have a distance within

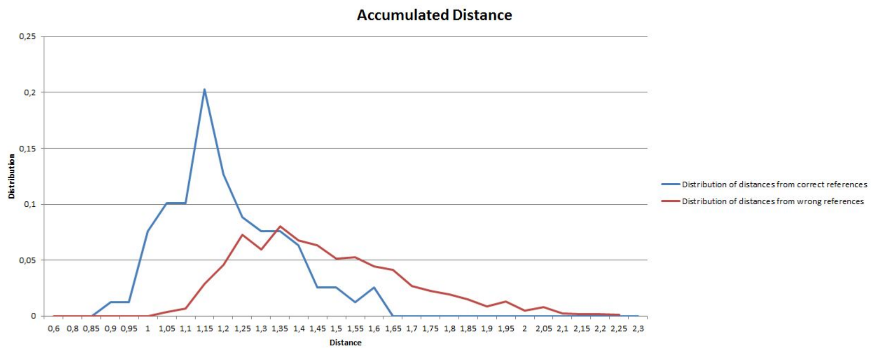

Figure 6.2: Distributions for the accumulated distance

0.85 and 1.65 from correct references, while the distances to wrong references vary between 1 and 2.1. Table 6.2 shows the FAR and FRR values when different thresholds are set for the accumulated distance.

Threshold	False acceptance rate	False rejection rate
1.25	15%	30%
1.30	21%	22%
1.35	29%	15%
1.40	36%	8%

Table 6.2: FAR and FRR for accumulated distance

The optimal distance threshold for this distance lies somewhere between 1.30 and 1.35. We see that the rates are slightly worse than the ones for the accumulated distance compared to the average step distance.

Average features distance

Figure 6.3 shows the distribution when the average features distance is used.

We see that this distance doesn't vary as much as the other two, but still has a significant overlap between the two distributions. Table 6.3 shows the FAR and FRR for different thresholds for the average features distance.

We see that the optimal threshold is somewhere around 0.30, creating a FAR and FRR on 22% and 15%, respectively.

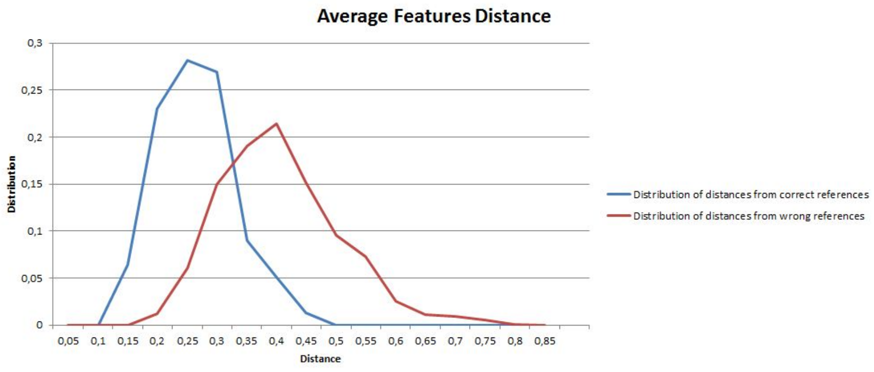

Figure 6.3: Distributions for the average features distance

Threshold	False acceptance rate	False rejection rate
0.25	7%	42%
1.30	22%	15%
1.35	41%	6%
1.40	62%	1%

Table 6.3: FAR and FRR for the average features distance

Tests on combinations of the values

A combination of optimal thresholds for the average step distance and the accumulated distance was first tested. A threshold for the average feature distance was also set, and was fixed to 0.45 to avoid an increase in the FRR when this threshold was used. Table 6.4 shows the results from this testing. The notations in the table are Avg for average step distance, Acc for accumulated distance and Fea for average features distance.

(Avg, Acc, Fea)	False acceptance rate	False rejection rate
(1.25, 1.35, 0.45)	18%	22%
(1.25, 1.30, 0.45)	16%	25%
(1.20, 1.35, 0.45)	12%	28%
(1.20, 1.30, 0.45)	10%	28%

Table 6.4: FAR and FRR for a combination of optimal thresholds for the accumulated distance and the average step distance

We see that the FRR is always above 20% when these thresholds are used.

An FRR over 20% can be considered very high. To be able to decrease the rate, an increase in some of the thresholds were needed.

As the previous results showed, the single distance that achieves the best rates is the average step distance. Therefore, only the threshold for this distance was set around the optimal value. The threshold for the other two was set around their maximum distance score observed for correct samples. By doing this, these two distances would not have a big impact on the FRR, while still limiting the FAR.

Table 6.5 shows the tests of different combinations and their corresponding rates.

(Avg, Acc, Fea)	False acceptance rate	False rejection rate
(1.25, 1.45, 0.45)	21%	15%
(1.25, 1.45, 0.40)	18%	15%
(1.25, 1.40, 0.45)	20%	16%
(1.25, 1.40, 0.40)	17%	16%
(1.25, 1.40, 0.35)	12%	18%
(1.20, 1.45, 0.45)	13%	23%
(1.20, 1.45, 0.40)	11%	23%
(1.20, 1.40, 0.45)	12%	23%
(1.20, 1.40, 0.40)	11%	23%
(1.20, 1.40, 0.35)	8%	26%

Table 6.5: FAR and FRR for different combinations of distances

We see that the optimal combination of thresholds is around (1.25, 1.40, 0.35), where the FAR and FRR are 12% and 18% respectively. The FRR has on average significantly decreased from the test shown in Figure 6.4.

Personal thresholds

We observe during the testing that people have different ways of speaking the correct sentence. Some often varies the way they speak the sentence, both in tone and length, while others have very little variations. The persons varying their way of speaking the sentence will typically have a higher average distance from their own reference than what persons with little variation will have. The task of the system will then be to spot these differences by looking at the training data for each individual, and set their individual thresholds thereafter.

As previously described in this chapter, the average step distance is used as the main classifying distance. This is therefore the distance that is focused on in this section. The results showed a good correlation between the average step distance from the correct reference from samples used in the training and the same distance for samples not used in training. Table 6.6 shows the average distance for samples trained on and samples not trained on for ten of the persons tested.

Person	Average step distance for samples trained on	Average step distance for samples not trained on
1	0.94	1.07
2	1.06	1.18
3	1.01	1.09
4	0.87	1.00
5	0.88	1.05
6	0.94	1.10
7	0.92	0.97
8	0.98	1.06
9	1.00	1.19
10	1.06	1.20

Table 6.6: Average step distance for trained and not trained samples

We see that the average step distance is typically a bit higher for samples not trained on. We also see that persons having a high average step distance for samples trained on also tend to have a high distance for samples not trained on. The same correlation can be seen for persons with low distances. This information can be used in the training process for setting personal thresholds.

The personal thresholds should not vary too much from the default threshold. Remember that the optimal threshold for the average step distance is around 1.25. It should not be possible to set this threshold to much more than 1.30, because then a significant part of wrong users will be accepted. Such a high threshold will create a better usability for users with a high average distance, but the FAR will be around 50%, creating a big security breach. At the same time, the threshold should not be set too low. The lowest should be around 1.10 to 1.15, since setting it lower will not increase the security with any significance. This is because there are very few illegitimate users who could achieve such low distances. The limits set for each average step distance are shown in Table 6.7.

Average step distance for samples trained on	Personal threshold
$distance > 1.05$	1.30
$1.00 < distance < 1.05$	1.25
$0.95 < distance < 1.00$	1.20
$distance < 0.95$	1.15

Table 6.7: Different thresholds for different distances

The system was then tested when the personal thresholds was implemented. The results for different limits for the accumulated distance and the average features distance are shown in Table 6.8.

(Acc, Fea)	False acceptance rate	False rejection rate
(1.40, 0.40)	13%	12%
(1.40, 0.35)	9%	17%

Table 6.8: FAR and FRR with the use of personal thresholds

We see that both the FAR and FRR are significantly improved from the use of default thresholds. The best rates are now around 13% for the FAR and 12% for the FRR. The reason for the improved FRR is that we now are able to accept many of the samples that previously were rejected for the users who typically got a score between 1.25 and 1.30. At the same time we are able to avoid any decrease in the rate for users that get a threshold lower than 1.25. We also see a distinct decrease in the FAR. This occurs because it is made possible to decrease the threshold for users who consistently get a score well below 1.25. Of the 16 persons tested on, ten got a personal limit of either 1.15 or 1.20. Only four of the users got an increased threshold. This means that on average the threshold has decreased from before. The calculated average threshold when personal limits are used is 1.21. A lower average threshold for the average step distance creates a lower FAR, as shown by the previous tests.

6.1.2 Identification

For the identification process, the distance classifier with the best performance was used. This is, as explained, the average step distance. Each sample was compared against all 16 references. The reference with the shortest average step distance from the compared sample was identified as the correct reference. Table 6.9 shows how many of their five samples that were correctly identified for each of the 16 persons in the test. Table 6.10 shows the calculated results from the data in Table 6.9. We see from the results that the calculated identification rate of the system is 81%.

Person	Amount of correctly identified samples
1	5
2	5
3	4
4	4
5	4
6	3
7	5
8	4
9	4
10	4
11	4
12	2
13	5
14	3
15	4
16	5

Table 6.9: Amount of correctly identified samples

Correctly identified samples	65
Total amount of samples	80
Identification rate	**81%**

Table 6.10: Identification results

6.2 Discussion

This chapter discusses the different results found in Section 6.1. In addition, it is discussed how well the system is able to provide the security and privacy requirements described in Chapter 2.

6.2.1 Authentication

Performance

One of the best trade-offs for the FAR and FRR are 13% and 12% respectively. Such high rates make the system not well suited for use in applications that are in need of a high security level. The use of the system would be more appropriate in applications as an alternative to the password-based authentication on smartphones.

We see that there is a distinct overlap between the results for correct and wrong samples for all the three distances tested. These overlaps create quite high FAR and FRR values for all three. We see that the performance for each of them is quite equal, with the average step distance having a slightly better performance. By combining the three distances we were able to improve the performance significantly.

One of the reasons for this improvement is the properties of the average step distance and the accumulated distance. Since both distances are derived from the same process, one would think the two distances would have a very high correlation. The experimentation and results showed that this is not always the case. In Chapter 5 it was shown that the two distances have two separate problems. The accumulated distance has problems with input samples of very short length, while the average step distance encountered the same problem with very long lengths. Even though the length of an incoming sample is limited to avoid this problem, it can still occur on a smaller scale. By using both these values in the decision process we can avoid many cases where one of these problems would create a false acceptance. In cases where the incoming samples are quite long, it could be possible that the calculated average step distance would be lower than the threshold. At the same time, the accumulated distance would typically be quite high in this case, because of the long length of the sample. We could then have the case where the average step distance is within the set threshold, but not the accumulated distance, making the system reject the sample. By combining the distances in the decision process we will therefore be able to avoid some false acceptances where only one of the two values is below the threshold.

The average features distance has quite a good performance. Its calculated FARs and FRRs are quite similar to the ones achieved using the average step distance. It is important though, that this classifier does not become too important in the classification process. The distance is a good classifier in regard to determining the correct voice, but is not capable of separating different sentences. Therefore, if the distance gets an important role in the classification, the system's ability to separate different phrases would decrease.

It is important to remember that the FAR and FRR values are calculated in a worse-case environment, where we assume that the attackers know the correct sentence. In a realistic environment this would not be true. Each user should keep their sentence confidential. The problem with this is that it gets very difficult to keep the sentence a secret when it is spoken each time the system is used. An attacker only needs to be close enough to the person to hear the phrase. Because of this we cannot totally rely on the secrecy of the passphrase.

Personal thresholds

The introduction of personal thresholds decrease both the FAR and FRR significantly. The reason for the decrease in the FRR was found by looking at the scores for a few of the test subjects. The results showed that some persons had quite a big variation in the distance calculated for each of their samples. When the default threshold was used, some of them had very few of their own samples within the threshold. This made their FRR very high, so that the system became unusable for them. By introducing personal limits the rate could be lowered significantly for these users.

The decreased FAR can also be explained by looking at the results from the same persons that caused the decrease in the FRR. These persons had, as mentioned, big variations in their results. Since the amount of samples in the testing was quite low, these variations had a big impact on how the default threshold was set. The threshold was therefore set higher than what was needed for many of the other test subjects. By using personal thresholds, these persons were able to get a lower threshold set, meaning that fewer attacks were made possible. This can be seen by looking at the average threshold, calculated to 1.21, lower than the default threshold of 1.25.

It is very important to not allow too big variations in this personal threshold. If a user has very big variations in the result, the personal threshold would be set very high to compensate for this. A high threshold will increase the FAR significantly. With a very high FAR the system could be deemed unusable if any level of security is needed. This is the reason for not enabling very big variations in the setting of the threshold.

The challenge when using such personal thresholds is to determine how the limit should be set for each person. How the limits are set in the system was decided by analyzing the test results. It is quite possible that these limits are not optimal in a larger scale. More tests should be performed in order to give more insights into this.

Limitations in the testing

The amount of samples recorded from each person was 15 in the test, which is not enough to get conclusive results. Since ten were used for training, we only had five samples from each person to evaluate the performance. This meant that the total amount of test samples were only 80. Such a low amount made the test results very sensitive to bad samples. One single bad sample could affect the FAR or FRR with more than one percentage point. It is believed that the system performance could be better than the results actually calculated here.

6.2.2 Identification

As shown in Chapter 6, the identification rate of the system is calculated to be about 81%. It is important to understand that this rate only apply for the case where the system contains the amount of persons the tests included, namely 16.

The system's performance with regard to identification can be compared against other similar systems, for example the system studied in [23], where several different techniques for speaker recognition are discussed. Both LPC and DTW are among the compared algorithms. The system in [23] achieves identification rates between 70% and 91% for different combinations of the algorithms. Their tests are done on 12 people, where 10 samples from each are used in a training procedure. The tests have used 141 samples in the actual testing. In comparison, our system's identification rate of 81% can be considered quite good when our tests included 16 people instead of only 12. It is important to note that the samples in the compared system were short, around 2 seconds, and recorded from noisy telephone speech. This in contrast to our system, where samples were a bit longer, and recorded in environments with small amounts of noise.

6.2.3 Security

We have not applied a data encryption technique in the system. However, a certain level of confidentiality of the data is achieved due to the fact that there are no recorded audio data saved in the system. The only thing stored are the extracted feature sets.

The system runs no direct integrity check on the incoming voice data. This means that the system does not know if the incoming sample originates from a recording or from live speaking. The problem for an attacker trying to use a recording to get access, is that he needs not only to have a recording of the correct voice, but the correct sentence as well. If the system was text-independent it would be possible to use any recording as long as it was of the correct voice. In this text-dependent system this will not be possible.

One way of improving the integrity check significantly, is to use a challenge-response technique. The technique makes the user enroll several different sentences in the system, and prompts one at random when the user wants to be authenticated. This would mean that an attacker would need recordings of all the different sentences. The big drawback with this method is that many users would quickly find it tiresome to provide training samples of many different sentences. It is therefore a good chance that the usability of the system would degrade with such an implementation. The question then becomes what's most desired, an increase in security, or no decrease in usability.

6.2.4 Privacy

The only sensitive information stored on each user in the system is the features of their voice. Each voice sample input to the system is deleted after the features are extracted. The features are stored in plain-text. Optimally these should be encrypted so that no sensitive information is visible.

There exist several cryptographic techniques that can be used to encrypt biometric references. Two examples are a fuzzy commitment scheme, created by Juels and Wattenberg [24] and a fuzzy vault scheme, created by Juels and Sudan [25]. These are encryption techniques that are able to accept small variations in the input and still decrypt correctly. This makes them suited to be used for encryption of biometric features. However, the ability to accept such variations typically makes them less secure than standard encryption techniques. As an example, the article "Finding the original point set hidden among chaff", by Cheng et al. [26], shows several attacks that can be performed on the fuzzy vault scheme.

We are not able to use the fuzzy commitment scheme to encrypt the features stored in this system. The scheme is built for encrypting single vectors of fixed length. In our system, the order and the length of the features typically vary. This makes it very difficult to use this encryption scheme in this system.

The fuzzy vault scheme is able to encrypt a set of values, which can have a different order. If an incoming set overlaps enough when compared to the reference set, it is decrypted correctly. The problem with using this scheme is that for two points in the two sets to be considered equal, they would have to be exactly equal. This is very rarely the case with the different feature vectors extracted from voice samples. These consist of float numbers, and almost always have some differences from sample to sample. Therefore we are not able to use this encrypt scheme either.

It might be possible to encrypt the references in the system, but a suited encryption technique has not been found so far.

Chapter 7

Conclusion and Future Work

This chapter summarizes the most important findings, and gives a conlusion on the work done in the thesis. Future work that can be done on the application is also proposed.

7.1 Conclusion

A text-dependent speaker recognition application for the Android platform has been made. The application makes use of several algorithms in the different phases. Most notably, LPC has been used for the main feature extraction, and DTW has been used for the comparison of features.

A training procedure was implemented to enable the use of more than one voice recording in the enrollment process. The implementation used the DTW algorithm to be able to align the features from the different voice recordings. The training procedure makes it possible to make a reference for the user which is close to his average utterance of the passphrase. It was shown that the use of the training procedure created a more stable performance in the authentication phase for users.

Personal thresholds were introduced to be able to set different thresholds for each user. Testing showed that the use of such thresholds significantly lowers both the FAR and FRR, and therefore improves the system performance.

Tests have been performed on the system in regard to both identification and authentication. The identification rate has been calculated to 81%, when the identification is restricted to distinguishing between 16 people. The FAR and FRR have been calculated to 13% and 12%, respectively. These rates can be considered too high for use in applications that are in need of a high level of security.

7.2 Future work

The application has several different features that can be implemented or improved. This section will discuss the ones that are considered most important.

7.2.1 User interface

During the implementation the focus has been put on providing a good performance of the voice recognition. In order to achieve as low as possible FAR and FRR, which are considered the crucial criteria of the performance, many tests have been performed and thereafter analyzed. Therefore, the interface made, mainly serves the purpose of testing the application.

All functionality for recording audio, training and authentication is already accomplished. A more user friendly interface will surely bring a new level of convenience and result in a better user experience.

7.2.2 Testing

More tests should be performed on the system to better analyze its performance. As discussed in Chapter 6, the amount of samples used in the testing was not enough to get conclusive results. New tests to be performed should consider this, and make test subjects record more samples. Therefore, one would have more samples from each test subject that can be used for the actual testing. It should also be considered to increase the amount of test subjects. This would give more insights with regard to the system performance over a larger set of subjects.

7.2.3 Preprocessing

The main testing done on the system has all been done in environments with small amounts of background noise. The quick tests done with varying amounts of noise and varying input sources have shown that the system is not able to handle many such variations very well.

To be able to perform well in noisy environments, the system must be able to remove the noise and other variations before the features of an incoming sample are extracted. Therefore, it is the preprocessing stage in the system that needs to be improved. An example of an improvement would be to add a noise removal algorithm to the preprocessing stage in the pipeline shown in Figure 3.2.

Bibliography

[1] State of the appnation - a year of change and growth in u.s. smartphones. http://blog.nielsen.com/nielsenwire/online_mobile/state-of-the-appnation-%E2%80%93-a-year-of-change-and-growth-in-u-s-smartphones/, May 2012.

[2] S. Mokhov, I. Clement, S. Sinclair, and D. Nicolacopoulos. Modular audio recognition framework. *Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada*, 2003, 2002.

[3] Voicevault. http://www.voicevault.com, June 2012.

[4] Verispeak embedded sdk. http://www.neurotechnology.com/verispeak-embedded.html, June 2012.

[5] A.K. Jain, A. Ross, and S. Prabhakar. An introduction to biometric recognition. *Circuits and Systems for Video Technology, IEEE Transactions on*, 14(1):4–20, 2004.

[6] ISO/IEC. *ISO/IEC 24745. Biometric Information Protection.* ISO/IEC, 2011.

[7] J.P. Campbell Jr. Speaker recognition: A tutorial. *Proceedings of the IEEE*, 85(9):1437–1462, 1997.

[8] Android. http://www.android.com, June 2012.

[9] Americaâs new mobile majority: a look at smartphone owners in the u.s. http://blog.nielsen.com/nielsenwire/online_mobile/who-owns-smartphones-in-the-us/, May 2012.

[10] E. Chin, A.P. Felt, K. Greenwood, and D. Wagner. Analyzing inter-application communication in android. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pages 239–252. ACM, 2011.

[11] InterNational Committee for Information Technology Standards. Study report on biometrics in e-authentication, 2007.

[12] WP3. Fidis deliverable d3.2: A study on pki and biometrics, 2005.

[13] WP3. Fidis deliverable d3.10: Biometrics in identity management, 2007.

[14] Julius O. Smith. *Spectral Audio Signal Processing.* `http://ccrma.stanford.edu/~jos/sasp/l`, 2012. Online Book.

[15] J. Makhoul. Linear prediction: A tutorial review. *Proceedings of the IEEE*, 63(4):561–580, 1975.

[16] H. Sakoe and S. Chiba. A dynamic programming approach to continuous speech recognition. In *Proceedings of the Seventh International Congress on Acoustics*, volume 3, pages 65–69, 1971.

[17] Z.M. Kovacs-Vajna. A fingerprint verification system based on triangular matching and dynamic time warping. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(11):1266–1276, 2000.

[18] N.V. Boulgouris, K.N. Plataniotis, and D. Hatzinakos. Gait recognition using dynamic time warping. In *Multimedia Signal Processing, 2004 IEEE 6th Workshop on*, pages 263–266. IEEE, 2004.

[19] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 26(1):43–49, 1978.

[20] Koders.com dtw algorithm. `http://www.koders.com/java/fid647E612AEB89260F5C952B17879C93F00A21CD80.aspx?s=AddOpts`, June 2012.

[21] Java sound programmer guide. `http://docs.oracle.com/javase/1.5.0/docs/guide/sound/programmer_guide/contents.html`, June 2012.

[22] Apache commons io. `http://commons.apache.org/io/`, June 2012.

[23] A. Ouzounov. Cepstral features and text-dependent speaker identification a comparative study. *Cybernetics and Information Technologies*, 10(1):3–12, 2010.

[24] A. Juels and M. Wattenberg. A fuzzy commitment scheme. In *Proceedings of the 6th ACM conference on Computer and communications security*, pages 28–36. ACM, 1999.

[25] A. Juels and M. Sudan. A fuzzy vault scheme. *Designs, Codes and Cryptography*, 38(2):237–257, 2006.

[26] E.C. Chang, R. Shen, and F.W. Teo. Finding the original point set hidden among chaff. In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pages 182–188. ACM, 2006.