DEGREE PROJECT IN COMMUNICATION SYSTEMS, SECOND LEVEL
STOCKHOLM, SWEDEN 2014

Seamless Speaker Recognition

ANARGYROS CHATZARAS
and
GEORGIOS SAVVIDIS

KTH ROYAL INSTITUTE OF TECHNOLOGY
INFORMATION AND COMMUNICATION TECHNOLOGY

Seamless speaker recognition

Anargyros Chatzaras
Georgios Savvidis

Master of Science Thesis

Communication Systems
School of Information and Communication Technology
KTH Royal Institute of Technology
Stockholm, Sweden

20 January 2015

Examiner: Professor Gerald Q. Maguire Jr.

Abstract

In a technologically advanced society, the average person manages dozens of accounts for e-mail, social networks, e-banking, and other electronic services. As the number of these accounts increases, the need for automatic user identification becomes more essential. Biometrics have long been used to identify people and are the most common (if not the only) method to achieve this task.

Over the past few years, smartphones have become frequently used gadgets. These devices have built-in microphones and are commonly used by a single user or a small set of users, such as a couple or a family. This thesis uses a smartphone's microphone to capture user's speech and identify him/her. Existing speaker recognition systems typically prompt the user to provide long voice samples in order to provide accurate results. This results in a poor user experience and discourages users who do not have the patience to go through such a process. The main idea behind the speaker recognition approach presented in this thesis is to provide a seamless user experience where the recording of the user's voice takes place in the background.

An Android application is developed which silently collects voices samples and performs speaker recognition without requiring extensive user interaction. Two variants of the proposed tool have been developed and are described in depth in this thesis. The open source framework Recognito is used to perform the speaker recognition task. The analysis of Recognito showed that it is not capable of achieving high accuracy especially when the voice samples contain background noise. Finally, the comparison between the two architectures showed that they do not differ significantly in terms of performance.

Keywords: *speaker recognition, user authentication, seamless operation, biometrics, standalone, client-server, Android.*

Sammanfattning

I ett teknologiskt avancerat samhälle så hanterar den genomsnittliga personen dussintals konton för e-post, sociala nätverk, internetbanker, och andra elektroniska tjänster. Allt eftersom antalet konton ökar, blir behovet av automatisk identifiering av användaren mer väsentlig. Biometri har länge använts för att identifiera personer och är den vanligaste (om inte den enda) metoden för att utföra denna uppgift.

Smartphones har under de senaste åren blivit allt mer vanligt förekommande, de ger användaren tillgång till de flesta av sina konton och, i viss mån, även personifiering av enheterna baserat på deras profiler på sociala nätverk. Dessa enheter har inbyggda mikrofoner och används ofta av en enskild användare eller en liten grupp av användare, till exempel ett par eller en familj. Denna avhandling använder mikrofonen i en smartphone för att spela in användarens tal och identifiera honom/henne. Befintliga lösningar för talarigenkänning ber vanligtvis användaren om att ge långa röstprover för att kunna ge korrekta resultat. Detta resulterar i en dålig användarupplevelse och avskräcker användare som inte har tålamod att gå igenom en sådan process. Huvudtanken bakom den strategi för talarigenkänningen som presenteras i denna avhandling är att ge en sömlös användarupplevelse där inspelningen av användarens röst sker i bakgrunden.

En Android-applikation har utvecklats som, utan att märkas, samlar in röstprover och utför talarigenkänning på dessa utan att kräva omfattande interaktion av användaren. Två varianter av verktyget har utvecklats och dessa beskrivs ingående i denna avhandling. Öpen source-ramverket Recognito används för att utföra talarigenkänningen. Analysen av Recognito visade att det inte klarar av att uppnå tillräckligt hög noggrannhet, speciellt när röstproverna innehåller bakgrundsbrus. Dessutom visade jämförelsen mellan de två arkitekturerna att de inte skiljer sig nämnvärt i fråga om prestanda.

Nyckelord: *talarigenkänning, användarautentisering, sömlös drift, biometri, fristående, klient-server, Android.*

Acknowledgements

We would like to sincerely thank our supervisor professor Gerald Q. Maguire Jr. for his continuous support and guidance. His accurate feedback helped us improve the content of our thesis and always pointed us towards the right direction. His experience and deep knowledge along with his enthusiasm are an inspiration for every student who works with him.

We deeply thank our families and friends for their constant support throughout our studies. Special thanks to our colleague and friend Edvald Eysteinsson for his help with the Swedish translation of the Abstract section of our thesis.

Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

API	Application Programming Interface
CMU	Carnegie Mellon University
DCT	Discrete Cosine Transformation
DFT	Discrete Fourier Transformation
EU	European Union
FFT	Fast Fourier Transformation
FM	Frequency Modulation
GMM	Gaussian Mixture Model
GMM-UBM	Gaussian Mixture Model-Universal Background Model
GPS	Global Positioning System
GSM	Global System for Mobile
IMEI	International Mobile Station Equipment Identity
JSTK	Java Speech Toolkit
JVM	Java Virtual Machine
LGPL	Lesser General Public License
LPC	Linear Predictive Coding
LPCC	Linear Predictive Cepstral Coefficients
LPCM	Linear Pulse-Code Modulation
MFCC	Mel Frequency Cepstral Coefficients

NIST	National Institute of Science and Technology
OS	Operating System
PDA	Personal Digital Assistant
SPEAR	Speaker Recognition
STT	Speech To Text
TED	Technology, Entertainment, Design
VQM	Vector Quantisation Model
VXML	VoiceXML
WAV	Waveform Audio File Format
W3C	World Wide Web Consortium
XML	Extensive Markup Language

Chapter 1

Introduction

This chapter describes the specific problem that this thesis addresses, the context of the problem, the goals of this thesis project, and outlines the structure of the thesis.

1.1 Overview

Today, nearly everyone owns a smartphone or a tablet computer. These types of devices have gained popularity mainly due to two reasons: they provide users with a wide range of services that simplify their daily lives while also their typically small size (compared to a desktop computer or a laptop) allows high portability and facilitates transportation.

To better explain what a smartphone is, we start with a brief retrospective. Initially, cell phones were created that allowed phone calls and (a bit later) text message exchange. Later Personal Digital Assistants (PDAs) were introduced, which were simply digital organisers before gaining wireless network connectivity. Over time, cell phones began adding PDA-like functions and vice versa, resulting in what is known today as a smartphone. In other words, smartphones can be perceived as cell phones with PDA features or as PDAs with cell phone features [1, 2].

A typical smartphone has, among other components, a touchscreen, camera(s), speakers, microphone(s), accelerometer, WiFi and Bluetooth interfaces, Global Positioning System (GPS) receiver and a relatively powerful processor. These components allow information about the user and the device to be retrieved, and create useful applications and accessories. For instance, this information could include the user's location, whether or not the user is facing the screen or is

holding the device close to the ear, the device's orientation and much more.

As already mentioned, microphone is a standard component of any smartphone. This gives us the opportunity to capture the user's speech and, after processing the sample through specific algorithms, extract features that allow us to answer questions such as "Who is speaking?", "What is being said?", and "What is the emotional state of the speaker?". One way of doing this, is by comparing the extracted features against previously stored features and recognise words, phrases and actual speakers [3].

1.2 Problem definition

Most of today's user authentication systems require a valid username and password combination. However, this type of systems does not identify the actual user. Anyone who knows a valid username and its corresponding password, can identify himself/herself, even if these credentials belong to someone else. The solution is to address this problem by using information that is unique for each user. This can be achieved by analysing user's biometrics, such as his/her voice.

In an attempt to address this issue, several manufacturers and individual researchers have developed *speaker recognition tools* that identify a user by examining the unique characteristics of his/her voice. In order to provide accurate results, most of these tools need to be trained by collecting a fairly large number of voice samples. This is typically done by prompting the user to read loud a relatively long text while recording his/her voice. This scenario results in a poor user experience and discourages users who do not have the patience to go through such a process.

1.3 Goals

The main goal of this project is to develop an application that performs speaker recognition with high accuracy, while at the same time minimising the required explicit user interaction. To achieve this, the application gathers samples of the user's voice while running in the background, *without* requiring the user to interact with the application. The collected data are used to continuously train the system with the aim to provide as accurate results as possible.

The expected outcome is a system where a user will be accurately identified after speaking shortly (for instance, after saying a single word), while at the same

time an illegitimate user will be rejected.

1.4 Scope

The scope of this thesis project is an Android application that seamlessly collects user speech samples and accurately performs speaker recognition. Enhancements regarding security issues are left open for implementation. This thesis describes a generic tool for speaker identification that can be extended to implement concrete services such as device personalisation based on the identified user's profile.

1.5 Target groups

This project might interest mobile application developers who are looking for alternative user identification solutions. It could also be interesting to enterprises which provide services that require user identification. For example, banks that offer e-banking services to their customers. This thesis contributes to research on voice analysis and voice feature extraction and comparison.

1.6 Structure

The rest of this thesis is organised as follows. Chapter 2 explains fundamental concepts used by this project, while also existing solutions related to speaker identification are presented. Chapter 3 describes the method used in this thesis. Chapter 4 gives a thorough description of the project's technical parts. In Chapter 5, the outcome of the conducted experiments is presented and analysed. Finally, in Chapter 6 we discuss the results of this project and describe what remains undone and could be implemented in a follow-up project.

Chapter 2

Background

This chapter explains fundamental concepts that are necessary for the reader to understand this thesis. It also presents existing tools and solutions related to speaker identification.

2.1 Uniqueness of human voice

The sound of each person's voice is considered to be unique because of a set of characteristics that can be grouped into *anatomical* and *behavioural* characteristics [4]. The human voice is the result of air passing from the lungs through the vocal track which is composed by the laryngeal pharynx (beneath the epiglottis), the oral pharynx (behind the tongue, between the epiglottis and the velum), the oral cavity (forward of the velum and bounded by the lips, tongue, and palate), the nasal pharynx (above the velum, rear end of nasal cavity) and the nasal cavity (above the palate and extending from the pharynx to the nostrils). This results in different vocal characteristics in terms of tone, frequency and range. Differences in pitch are noticeable between adult men and women, as a result of differences in larynx size of the two genders.

Behavioural characteristics also affect the human voice and may change over time. Motion of the mouth, pronunciation and emotional state are some of the behavioural components of a person's voice [5]. This type of biometrics is also known as *behaviometrics* [6].

Establishing the individuality of someone's voice, can be a very difficult task considering the large world population. Naresh et al. [7] have proved this assumption by using a statistical model, namely *statistically inferable dichotomy model*, which has previously been used to establish the uniqueness of handwriting

and of fingerprints. This model allows us to experiment with a small group of people and safely generalise the results to the entire population.

2.2 Identification, authentication and authorization

Three commonly confused concepts in information security are *user identification, user authentication* and *user authorisation*. While identification is, basically, identifying a single user among a finite number of users, authentication is the act of proving that a user is indeed who he claims to be. For example, in a speaker recognition system, identification would be the process of finding which user, from a certain group of registered users, has just spoken. Authentication would be the process of proving that a speaker who claims to be John, is indeed John.

Authorisation, on the other hand, takes place after a user has been authenticated. It is the process of defining what the authenticated user is allowed by the system to do. For instance, if the user is an administrator, then he/she has probably more privileges than a regular user in the system [8].

2.3 Speech recognition and speaker recognition

Speech recognition and *speaker recognition* are two completely different technologies. Speech recognition refers to the process of translating spoken words and phrases into text and is also known as Speech To Text (STT). It has become a widely used technology in automated systems in sectors such as military, aerospace, education, health care but also in daily life services. Google's *Google Now* and Apple's *Siri* are two common examples of services that use speech recognition technologies.

Speaker recognition, on the other hand, is the process of determining *who* the actual speaker is instead of what he/she has said. Speaker recognition depends on biometrics, while speech recognition does not. In this thesis, we focus on identifying the speaker (speaker recognition) instead of determining what a speaker has said (speech recognition).

2.4 Speaker recognition process

Speaker recognition systems consist of two phases, namely *training phase*, also known as enrolment phase, and *recognition phase*, also known as testing phase. The recognition phase cannot take place without first having initiated the training

phase. However, this does not mean that the training phase needs to be completed before recognition can be performed. In fact, the training phase can be an ongoing process, running simultaneously with the recognition phase. In particular [3, 5]:

Training phase
: During this phase, the system is trained to recognise a speaker. The training set consists of voice recordings that may vary from a single word to several hours speech. The longer the training phase, the more information the system learns about the user's voice and, thus, the more accurate it becomes. The outcome of this phase is a *model*, which describes a user's voice.

Recognition phase
: During this phase, the system attempts to recognise an initially unknown speaker, based on stored models. After capturing a speech sample from this unknown user, the system compares the characteristics of this user's voice to the stored models. The result of this comparison is a likelihood percentage that will be used to determine whether or not the unknown user is in fact one of the known, registered users. The minimum value of this certainty threshold for a match to be considered valid, and thus for the unknown user to be recognised as a registered user, may vary from implementation to implementation.

In many speaker recognition systems, the training phase is a fixed step that is finalised once the model is created and before the recognition phase starts. In this project, however, the training phase is an ongoing procedure and does not terminate after the model for a user is created. A user's model is continuously improved based on new input that is captured over time. Speech samples used by the recognition phase can also be used by the training phase to improve an existing model. If during the recognition phase, the recordings of an unknown user match an existing model, then these recordings can be used to further improve this model.

The performance of the system is affected by a variety of factors that can be briefly grouped in the following three categories for this thesis:

Technical factors
: For example, at which rate is the voice sampled.

Recording environment factors
: For instance, the distance between the speaker and the microphone, the sound level of the user's voice, etc.

Speaker physiological and psychological factors
: For example, a cold, stress, etc.

For the speaker identification task, only the voice should be considered. This means the voice of the user needs to be loud and that any background noise must be removed by applying some noise cancellation filters. The National Institute of Science and Technology (NIST) has published a plan targeting the evaluation of speaker detection systems [9].

2.5 Speaker recognition types

There are two types of speaker recognition systems, depending on the content of the speech that the user gives to the system [3, 5]:

Text dependent
: Also known as constrained mode. During the training phase, the user speaks a utterance that can vary from a single word to a relatively long phrase. During the recognition phase, text dependent systems require the user to say the same word or phrase that he/she gave as an input during the training phase.

Text independent
: Also known as unconstrained mode. Unlike text dependent systems, in text independent systems, the user is not required to speak the same utterance during both the training and the recognition phase. Usually in this kind of systems, the training phase is longer than it is in text dependent systems. In addition, a text independent system might require a longer utterance during the recognition phase in order to accurately recognise the speaker.

2.6 Voice features

In the training phase, the user registers himself/herself to the system, by providing a number of samples of his/her voice. These samples (in the case of text-independent voice recognition) are split into smaller frames of 20-30 milliseconds. These frames are then processed in an effort to suppress any channel impairments [4] and extract the unique characteristics of human voice (tone, speed, pitch, etc). A variety of characteristics, called *features*, are extracted from the voice sample and compose a *voiceprint*. The features that one could extract from a sample, can be grouped in five categories [10]:

- Short-term spectral features, extracted by performing Discrete Fourier Transformation (DFT) or other, more complicated transformations on the samples.

- Voice Source Features, which are dependent on anatomic uniqueness of user voices (e.g. on the speaker's glottal pulse shape).
- Spectro-temporal features, which by analysing the samples in the frequency domain, can capture details caused by intonation. A representative of this type of features is Frequency Modulation (FM) Features.
- Prosodic features, such as the speaker's rhythm, or emphasis on a specific sound (e.g. syllable emphasis due to a dialect), and are expressed by the fundamental frequency (F0).
- High-level features, which focus on the finite set of words that a person usually uses (idiolect).

2.6.1 Feature extraction techniques

Several feature extraction techniques are used in the field of audio and speech processing. This section briefly presents two of the most popular ones, Linear Predictive Coding (LPC) and Mel Frequency Cepstral Coefficients (MFCC).

2.6.1.1 Linear Predictive Coding (LPC)

LPC is one of the most powerful speech analysis methods while it is also very useful for encoding high quality speech at low bit rate. The main idea behind LPC is that a future speech sample can be predicted by linearly combining past speech samples [11]. From a given audio sample, LPC extracts *Linear Predictive Cepstral Coefficients* (LPCC) [12], a common set of features for speech processing [13]. The basic steps of a typical LPC processor are [14]:

Pre-emphasis
: The digital sound signal is flattened in order to become less susceptible to *finite precision effects*.

Frame blocking
: The outcome of the previous step is broken into frames of (typically) 20 milliseconds long [11].

Windowing
: Each frame is passed through a window function in order to decrease gaps at the beginning and the end of the signal.

Autocorrelation
: After windowing, each frame is autocorrelated.

LPC analysis
The derived autocorrelation values are converted into an LPC parameter set by implementing the *Levinson-Durbin recursion* method.

LPCC extraction
Finally, the LPC parameter set is converted into LPCC.

2.6.1.2 Mel Frequency Cepstral Coefficients (MFCC)

MFCC is one of the most commonly used techniques for feature extraction [13]. The main disadvantage of MFCC is its sensitivity to the presence of noise because of its dependence on the spectral form [11]. Tyagi and Wellekens [15] have proposed modifications to the MFCC algorithm in order to improve noise robustness. The main steps for the MFCC derivation are [13]:

Pre-emphasis
The digital sound signal is flattened in order to become less susceptible to *finite precision effects*.

Frame blocking
The outcome of the previous step is blocked into frames. In order to smoothly transit from frame to frame, overlapping of the frames is often used.

Windowing
Each frame is passed through a window function in order to decrease discontinuities at the beginning and the end of the signal.

Fast Fourier Transformation (FFT)
After windowing, FFT is derived for each frame in order to extract frequency components.

Mel scale filter bank
The Mel scale filter bank is applied to the outcome of the frame's FFT. The scale is approximately linear for frequencies up to 1 kHz and logarithmic for frequencies above 1 kHz. The reason for this is that the human ear becomes less frequency-sensitive as the frequency increases above 1 kHz [11].

Log
The logarithm of the Mel scale filter bank output is computed.

MFCC extraction
Finally, the MFCCs are derived by calculating the Discrete Cosine Transformation (DCT) of the previous step's outcome.

2.6.2 User models

After the feature extraction phase, the voiceprints are stored as *user models* in a database. A user model can be described as a data structure that describes a user. For example, a simple user model could consist of a voiceprint and a username. The decision of which features will be used, affects how the *pattern-matching* between the original voice sample and the sample that needs to be identified, will be addressed. Also, depending on which features are used, an appropriate model should be chosen. Different types of models perform better for specific features than others. Although using all the available features would theoretically maximise the system's accuracy, it is practically impossible to do so since larger feature sets result in larger models. The size of the model affects the computation delay of the pattern matching step [4].

The types of models can be categorised into *stochastic* (parametric) and *template* (non-parametric) models. In stochastic models, each speaker is *"modelled as a probabilistic source with an unknown but fixed probability density function"* [10]. Estimating the parameters involved in this density function from the samples provided composes the training phase for this type of models. In stochastic models, the pattern-matching problem is expressed as a probability of matching two samples. A commonly used type of stochastic model is Gaussian Mixture Models (GMM) [16]. In template models, the pattern-matching problem is much simpler and is solved by calculating the distance between the two samples. A typical template model is the Vector Quantisation Model (VQM) [17].

Regardless of what kind of model is used, a sample can be affected by recording-specific factors, as described previously. The possible errors are either falsely identifying a fake user or not identifying a legitimate one. Mismatch in the quality of these factors can highly affect the accuracy of the system. Thus, a necessary step in order to make the system more robust is to try to eliminate this mismatch by normalising the "middle step products" in the process. Examples include feature normalisation, speaker model compensation and score normalisation [10]. For a detailed description of the challenges of fine-tuning the decision threshold, the reader may refer to Furui [18] and Bimbot et al. [19].

2.7 Android and device identification

Android is one of the most popular mobile operating systems, powering more than one billion devices around the globe [20]. Initially created by Android Inc. and later bought by Google [21], Android Operating System (OS) is an open source

platform based on the Linux kernel and designed for touchscreen devices.

In Android, there exists a variety of ways to uniquely identify a mobile device. First, all devices are required to have at least one Gmail account. In addition to this, every Android installation later than Android Froyo 2.2, assigns a unique id, namely *ANDROID_ID*, to the current installation of Android itself. This is considered a secure and reliable token that uniquely characterises an Android device [22], even though the Google Team announced some issues at "a popular handset from a major manufacturer, where every instance has the same ANDROID_ID" [23]. If the device is a Global System for Mobile (GSM) enabled device, it is assigned an International Mobile Station Equipment Identity (IMEI) number. This allows tracking a unique instance of an application, running on a specific device. However, this approach excludes all non-GSM enabled Android devices, such as several tablets.

2.8 Related work

This section presents existing tools and research, related to the field of speaker recognition. In order to be able to evaluate the different methods in biometric research, a vocabulary of comparison has been developed. *False match rate* is the rate at which a system falsely identifies a fake user's sample, while *false non-match rate* is the rate at which a system fails to identify a real user. Both these errors push the acceptance threshold in different directions; failure to identify a valid user means the threshold is too high and needs to be decreased, while falsely identifying a fake user means that the acceptance threshold is too low and, thus, needs to be increased. The terms *correct match rate* and *correct non-match rate* are also used in this thesis. The first describes the rate at which a system accurately identifies a speaker while the latter describes the rate at which a system correctly does not identify a fake user. Table 2.1 summarises what these four metrics represent when comparing the system's output to reality.

Table 2.1: Speaker recognition accuracy metrics

Reality \ System	**Real user**	**Fake user**
Real user	Correct match rate	False non-match rate
Fake user	False match rate	Correct non-match rate

2.8.1 VoiceXML and speaker recognition

VoiceXML (VXML) is a digital document standard, developed by the World Wide Web Consortium (W3C). The format of a VoiceXML document is based on Extensible Markup Language (XML). The standard is designed for creating audio and interactive media dialogs between computers and humans. Its main goal is to add the advantages of web-based applications to interactive voice response applications, such as automated customer service portals. Figure 2.1 represents a simple VoiceXML document that synthesises and presents the voice message "Hello world!" to the user, after which, the conversation ends. The top-level <vxml>tag is a container for dialogs. Two types of dialogs exist: *forms* that display information and accept user input, and *menus* that offer choices of what the next step should be [24].

```
<?xml version="1.0" encoding="UTF-8"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
  <form>
    <block>
      <prompt>
        Hello world!
      </prompt>
    </block>
  </form>
</vxml>
```

Figure 2.1: Example of a VoiceXML document

The VoiceXML standard defines XML elements that instruct the interpreter client to provide speech synthesis, speech recognition, audio playback, etc. However, the standard does not provide support for speaker recognition. To achieve this, several third-party organisations have extended the standard, adding support for this functionality. A few of these approaches have been selected and are presented below.

BeVocal introduced an experimental extension to the VoiceXML standard in order to support speaker verification. This extension includes all the necessary XML elements for implementing the two phases of the speaker recognition procedure, training and recognition, as described in Section 2.4. Along with this VoiceXML extension, BeVocal released a guide describing how to use their approach to create the necessary dialogs for a speaker recognition application [25, 26].

IBM's WebSphere Voice Server, software for developing and deploying

conversational applications, provides a speaker verification component that allows VoiceXML-based applications to submit audio for verification by a server. This product has gained popularity among e-business applications that use audio dialog technologies [27, 28].

2.8.2 Recognito

Recognito [29] is an open source speaker recognition framework written in Java and developed by Amaury Crickx. Although it is still in an early development stage, Recognito is a promising project with its main advantages being simplicity and extensibility.

2.8.2.1 Feature extraction process

Recognito converts a voice sample into a voiceprint by using the LPC feature extraction method described in Section 2.6.1.1. Before the feature extraction phase, Recognito first removes silence since it is unnecessary overhead in the process and it then applies normalisation to the given voice sample, in order to make the feature extraction process unbiased towards sound volume sensitive features. The selected technique for removing silence is based on a characteristic of *white noise*: when applying autocorrelation to white noise, the average value of the derived coefficients is close to zero. Recognito removes chunks of silence when they are at least 4 seconds long and there has not been speech of 200 milliseconds or longer, during each chunk. Normalisation is performed in order to apply a constant amount of gain to the entire sample. The value of the applied gain is the maximum value of gain that exists in the sample before the normalisation process starts.

The extracted features are stored in an array of *double* data type with a length of 20 items. Given that in Java the size of a double is 8 bytes, this means that the amount of memory required to store the extracted features is approximately 160 bytes (in reality there are a few extra bytes of overhead due to the array's header). In addition to the extracted features, Recognito's voiceprint contains some general information, resulting in a size of approximately 350 bytes.

2.8.2.2 Training phase

During the training phase, Recognito simply *merges* the voiceprint of an existing user with a new voiceprint that was given as an input for this purpose. In particular, Recognito recomputes the mean values of the voice features of the existing voiceprint after adding the features of the new voiceprint. Recognito

provides convenient functions to train existing user models with new voiceprints. Recognito does not provide any logic regarding whether or not a new voiceprint is adequate for training a particular user model. Decisions like these, are implementation dependent.

2.8.2.3 Recognition phase

The recognition process is based on the so called *Universal Model*, which is an average of all known voiceprints (voiceprints that correspond to stored user models) in the system. This means that the Universal Model itself is actually a voiceprint. Each time a voice sample is sent to Recognito in order to recognise its speaker, a voiceprint is generated. For each stored voiceprint, Recognito calculates the Euclidean distance between this voiceprint and the Universal Model. It then calculates how close the unknown voiceprint is to each stored voiceprint, compared to the total distance between the stored voiceprint and the Universal Model. The outcome is a *likelihood ratio* that describes the certainty percentage that the unknown voiceprint belongs to the same user as the stored voiceprint. The shorter the distance between the unknown voiceprint and a stored voiceprint, the higher the likelihood that they both belong to the same speaker.

Since the Universal Model is the mean value of all the stored voiceprints, this means that if there is only one stored voiceprint, then the Universal Model will be equal to this voiceprint. Consequently, in such scenario an unknown voiceprint will always be equally close both to the Universal Model and the stored voiceprint, resulting in a 50% likelihood ratio, as shown in Figure 2.2. The more the stored voiceprints in the system, the more relevant the likelihood ratio becomes.

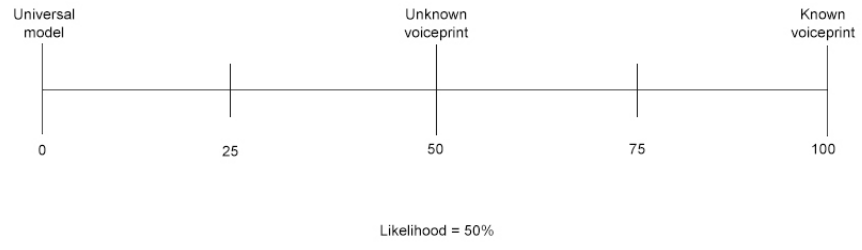

Figure 2.2: Recognito's likelihood ratio

2.8.3 Other speaker recognition systems

Cristian M. Toader [30] extended the Pico project, presented by Frank Stajano in [31], in order to create a multi-level unlocking model. Pico was originally designed as an alternative authentication mechanism that replaces passwords with authentication tokens. Pico unlocks after successfully communicating with small devices, called Picosiblings, that are embedded in everyday items, such as keys, jewellery, etc. Pico receives secret sequences by the Picosiblings and, when all the necessary secrets are collected, it becomes unlocked and ready for use. In his Master's Thesis, Cristian M. Toader addresses as the main downside of Pico the fact that it does not guarantee the identity of the owner of the Picosiblings. That is, anyone who posses the required Picosiblings can unlock Pico and gain full authentication privileges. His solution to this problem is a multi-level unlocking scheme that combines biometrics and behaviometrics, such as iris, face, voice and gait. To access these metrics, the role of the Picosibling is moved into a mobile application that runs on an Android device. Each of these metrics has a weight, based on the level of confidence it offers in identifying a person. In addition, each metric generates a probability that the actual owner possesses the device. The weighted sum of all these probabilities is then used to generate an overall confidence level which is, in turn, used to unlock applications that are associated with a confidence level that is lower than the generated one. For the voice recognition process, Pico uses the Recognito library [29].

In [3], Carlos Dominguez Sanchez developed a speaker recognition tool for handheld computers. The tool is written in C++ and it requires a single word to be used for the training phase and the same word for the recognition phase. Moreover, it uses the *Euclidean distance* to find the best match. In particular, suppose there are two stored models, *model A* and *model B*, that represent *user A* and *user B*, respectively. The total distance between *model A* and *model B* is the sum of the Euclidean distances between each feature of *model A* and the corresponding feature of *model B*. Thus, in order to find a match for an unknown user model, the distance between this model and all the stored models, has to be calculated. The shortest distance indicates the best match.

Alize [32] is an open source platform, distributed under Lesser General Public License (LGPL) license and developed by the University of Avignon that is used for voice and image verification. It is written in C++, and it offers a plethora of configuration options, regarding for example which model to use.

The Java Speech Toolkit (JSTK), a library developed by the University of Erlangen-Nuremberg, is used for speech recognition, speaker verification and

related tasks. It is written in Java is directly portable to Android, which means that some of the calculations can take place in the device, if needed [33].

Another popular solution is Carnegie Mellon University's (CMU) Sphinx, which is a solution originally for speech recognition, but its modular design allows for extension in order to perform speaker identification [34].

Other solutions in Python such as voiceid [35] and Speaker Recognition (SPEAR) [36] also exist, although Python is not officially supported by Android.

Given the fact that the task of speaker verification consists of computationally expensive operations, such as feature extraction and creation of models, systems that utilise a server-based approach have been proposed. In [37] a distributed system for speaker recognition is described which uses Gaussian Mixture Model-Universal Background Model (GMM-UBM) models. A similar approach has been described for speech recognition in [38]. Both proposals use a distributed architecture composed of a frontend and a backend. The frontend is responsible for removing noise in the sample and extracting the desired features, while the backend is responsible for all the rest.

Brunet et al. [39] proposed a method that performs the entire speaker recognition process on an Android mobile device. To achieve this, they extract the MFCC features, and store them as a distance vector. For the pattern-matching step, during the testing phase, they compare the test samples and extract their Euclidean distance. They used samples both from a publicly accessible and a private database, achieving promising accuracy results.

Chapter 3

Method

The *engineering method* and the *scientific method* are two different processes of obtaining human knowledge. While the scientific method is related to *understanding how things are*, the engineering method's purpose is *creating what has never existed* [40]. By making observations and conducting experiments, the scientific method aims at proving a hypothesis that explains a natural phenomenon. On the other hand, the main goal of the engineering method is to provide a solution to a known problem [41]. The steps of the scientific method can be summarised as follows:

1. Construct a hypothesis that explains a phenomenon.
2. Test the hypothesis with an experiment.
3. Analyse the results and validate the hypothesis. If the outcome does not align with the hypothesis, then repeat the process from step 1.
4. Communicate the analysis results.

The steps of the engineering method can be summarised as follows:

1. Define the problem. Do background research and identify what needs to be solved.
2. Specify requirements and constraints in order to provide quality results.
3. Develop a solution and verify that it meets the requirements. If not, then repeat this step.
4. Communicate the analysis results with regards to time constraints, energy consumption, etc.

In order to develop a seamless speaker recognition tool, not currently available in the market, this thesis follows the engineering method, slightly altered at step 3. Generally, a common way of working with the engineering method involves going back and forth between steps. This is called *iteration* and it is also adopted by this thesis.

First, we define the problem that needs to be solved. After in-depth background research, we identify what exactly has not been solved in the field of speaker recognition.

In the second step, in order to specify the requirements and constraints of our solution, we first have to fully understand how the speaker recognition process works and what types of speaker recognition exist. Moreover, since the application will be designed for the Android platform, we first study the platform's specifications, restrictions and guidelines. This step is tightly linked to the outcome of step 1.

For the third step, instead of one, we develop two different prototypes that we later compare to conclude which one of them meets the requirements set in the previous step. The differences between the two solutions are explained in detail in Chapter 4.

Finally, the two solutions, along with their evaluation results, are presented, while problems that remain unresolved or require further improvement are also discussed.

Chapter 4

A seamless speaker recognition mechanism

This chapter presents two different architectures of the proposed speaker recognition system. In addition, this chapter describes restrictions and limitations of both architectures. The actual speaker recognition procedure is performed by using the open source framework Recognito, which was described in Section 2.8.2.

4.1 Restrictions and assumptions

This section describes restrictions and assumptions that need to be taken into consideration in order to achieve a good user experience.

4.1.1 Audio recordings

In order to convert a voice sample into a voiceprint, the sample has to be loaded in the heap memory of the device that will perform this operation. The larger the audio file, the more likely it is for the device to run out of memory. To reduce this risk, while capturing user's voice we store the recordings in small chunks with maximum duration of 60 seconds each. The format of the recordings is summarised in Table 4.1.

Table 4.1: Voice samples format

File format	Waveform Audio File Format (WAV)
Encoding format	Linear Pulse-Code Modulation (LPCM)
Sample rate	44100 Hz
Bit depth	16 bits
Channels	1 (Mono)
Bit rate	1411.2 kbit/s

According to the official documentation, a sample rate of 44100 Hz is the only rate that is guaranteed to work on all Android devices. Also, single channel audio (mono) is the only channel configuration that is guaranteed to work on all Android devices [42]. In addition, Recognito performs better with mono channel audio. Using a stereo channel audio will double the processing time while it will also decrease the accuracy, if the two channels are not identical [29].

Using the above format means that an audio file with a duration of 60 seconds will be approximately 5.2 MB. The file size can be derived from the formula below:

```
Size = (SampleRate * Channels * BitDepth * DurationInSeconds) / (8 * 1024)
```

which results in:

```
Size = (44100 * 1 * 16 * 60) / (8 * 1024)
Size = 5.167 MB
```

4.1.2 Energy consumption

Having the device's microphone constantly enabled and recording user's speech, may result in unnecessary battery consumption while it might also cause recording failure for other applications if a device does not support multiple instances of the same codec [43].

In order to avoid this, the application periodically checks for voice signals (*capturing* state). If not detected, the application enters an *idle* state, releasing resources and consuming minimal battery energy. After this time has elapsed, the application will check for voice and if speech is detected, the application will

enter the *recording* state. If a phone call is established, the idle state will be instantly interrupted and the recording state will be entered, since it is very likely that the user will speak. If the battery level is lower than 10%, the application will become *inactive*. As soon as the device's charging state has changed to *charging*, the application will switch back to its capturing state. It is assumed that a mobile device of an average user is not used during night hours. Thus, in order to further reduce battery utilisation, the application will stop running between 23:00 and 08:00 local time, by default. During this period, the application enters again the inactive state. The user can configure this time range through the application's settings. Table 4.2 describes the possible states of the mobile application. Figure 4.1 illustrates the main application flow as long as the application is not in the inactive state.

Table 4.2: Application states

State	**Description**
Capturing	The application is capturing input voice in order to determine whether or not a user is speaking.
Recording	The recording begins when either a phone call has been established or voice has been detected during the capturing state. The recording state lasts for 60 seconds (not configurable by the user) when triggered through the capturing state or, in case it was initiated by a phone call, until the phone call is terminated.
Idle	No voice was detected during the capturing state. The application resources will be released for 60 seconds in order to reduce energy consumption. After this amount of time has elapsed, the application will go back to the capturing state. The duration of 60 seconds is the default value and can be changed by the user from the application's settings.
Inactive	There are two reasons the application can be inactive. The first reason is that it stopped running because it entered the defined nightly period. The second reason is that the battery level has gone below 10%. While inactive due to the second reason, the application only checks for the device's charging state. The default value of 10% can be changed by the user from the application's settings.

Finally, regarding the client-server architecture and, in particular, the user model's update at the server (as described in Section 4.2.1), instead of opening

a channel between the mobile device and the server every time there is a new voice sample recorded, recordings are temporarily stored on the device in the form of voiceprints and are sent to the server every 24 hours. This means that the transmissions take place nightly at 23:00 local time (unless specified otherwise by the user) or, in case of low battery or no network connectivity, the next available.

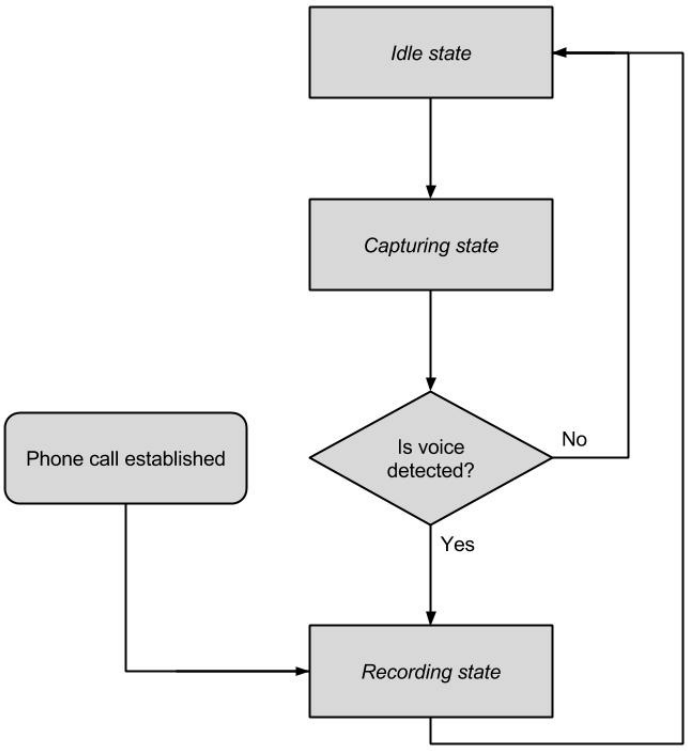

Figure 4.1: Main application flow

4.1.3 User experience

As mentioned before, the user models will be updated by voice samples captured in the background, without requiring user interaction. However, it is vital that the first voice sample that is used to create the corresponding user model, comes from the actual user and not from a random person that happens to speak close to the microphone during the first recording. Thus, in order to create a new user model, user interaction is required. To achieve this while maintaining a good user experience, the application will prompt the user to speak for approximately 10 seconds.

Another issue that needs to be addressed regarding the update of the user

models (training phase) is the certainty that the captured voice belongs to the same person as the one represented by the user model that is about to be updated. For instance, consider *user model A* that represents *user A*. In order to update this model, it is important to first make sure that the recorded voice belongs to *user A*. In other words, the following question needs to be answered: "What is the minimum acceptable certainty required to update a user model?". Section 2.8.2 described how Recognito calculates its likelihood ratio. For this thesis, we assume that it is safe to update a user model when the likelihood ratio is at least 75%. This threshold is referred to as *certainty threshold*. When the system receives a new voiceprint, it compares it against all stored voiceprints. If it reaches the certainty threshold for one of the stored voiceprints, it assumes that the new voiceprint belongs to the same user and consequently updates the corresponding user model. If the certainty threshold is reached for more than one stored voiceprints, then the system selects the one that generated the highest likelihood ratio and updates the corresponding user model accordingly.

4.2 Seamless speaker recognition architectures

In this section, two different architectures for the proposed speaker recognition mechanism are presented. The first approach is a centralised architecture where a server is responsible for carrying out part of the required speaker recognition process. The second approach is a standalone architecture where all the procedures and computations take place locally on each mobile device. Each approach provides several advantages and new features compared to the other and may be used to serve different use cases.

4.2.1 Client-server architecture

In the first scenario, we consider a centralised system where part of the process takes place on a server. In particular, the mobile device is only responsible for recording the user's voice, converting it into a voiceprint and sending the voiceprint to the server.

Initially, the user is prompted to select a username and to read loud a short text. The speech is converted into a voiceprint and along with the username is sent to the server in order to create a new user model. The server replies with a *success* message if the operation was carried out successfully or with a *failure* message if the username is already in use.

After this initial phase, no further user interaction is required for the training phase. The application runs in the background, switching between states, as described in Table 4.2. All recorded data are temporarily stored on the device in the form of voiceprints and are sent to the server once a day in order to update the corresponding user models. Figure 4.2 describes what the communication between the client and the server looks like. There are mainly two reasons for converting the voice samples into voiceprints on the mobile device and not on the server:

Data length

As explained in Section 4.1.1, a voice sample can be up to 5.2 MB. Sending multiple samples of this size from the client to the server may result in excessive network overload as well as increased delay for the serialisation and deserialisation of the transmitted data. This problem is solved by converting the voice samples into voiceprints and transmitting the latter. Section 2.8.2.1 mentions that the size of a voiceprint is approximately 350 bytes, significantly smaller than a 5.2 MB voice sample.

Privacy concerns

Storing voice samples and transmitting them through a network has serious implications in users' privacy. The risk of leaking private conversations is too high to overlook. This by no means implies that voiceprints do not contain sensitive information. With regards to privacy concerns, the main advantage of storing voiceprints instead of voice samples is that the *content* of a conversation cannot be reproduced. The identity of the speaker, though, might still be retrieved.

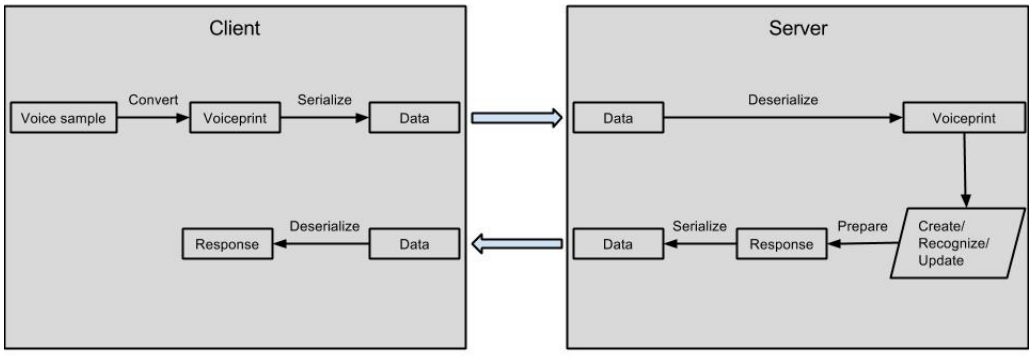

Figure 4.2: Client-server communication

In the client-server architecture, the recognition phase can occur in two ways:

Automatic recognition
The mobile device sends voiceprints to the server once a day (by default at 23:00) in order for the server to update the corresponding models. When the server receives a new voiceprint, it has to recognise the speaker in order to update the appropriate model. Once this is done, the server responds to the client with the updated user model. This process is called automatic recognition. However, sending voiceprints once a day means that there is a high probability that they belong to multiple registered users. In such case, the automatic recognition will not detect a single user. To solve this problem, along with the voiceprints sent to the server, the mobile device includes a timestamp that represents the creation time of each voiceprint. The server updates all the matched user models and, based on the timestamps, responds with the model that corresponds to the lastly created voiceprint. In other words, the server will respond with the model of the last of the registered users whose voices were recorded during the day.

Instant recognition
This option allows a registered user to be recognised at any time, instead of waiting for automatic recognition to be performed. If the user selects this option, he/she is prompted to speak shortly. The recording is converted into a voiceprint and is sent to the server. The server, in turn, recognises the user, updates the appropriate model and responds to the client with the user model.

Finally, each device has a set of authorised users and only these users can use the application on this particular device. The aim is to provide a certain level of control to the device holder over who is allowed to use the application. In addition, a user can be authorised to use the application on multiple devices. This is achieved by linking each user model with the ANDROID_ID(s) (as described in Section 2.7) of the device(s) on which he/she is allowed to use the application. Figure 4.3 illustrates a simple scenario where different users are authorised to use different devices. A user can be authorised to use the application on a particular device in two ways:

- He/she was granted access by an already authorised user. The latter needs first to be recognised by the device before authorising new users.
- If there are no authorised users on the device, the application prompts the user to authorise himself/herself either by registering to the system for the first time or by entering his/her existing username.

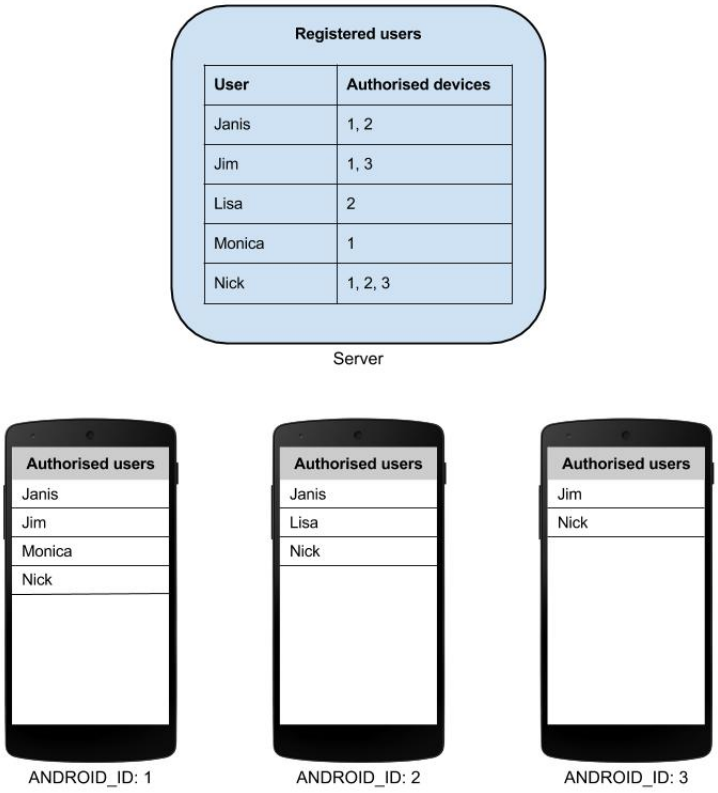

Figure 4.3: Example of authorised users per device

4.2.2 Standalone architecture

In the second scenario, we consider a standalone architecture without a server. Similar to the client-server approach, in order to create a new user model, the user is prompted to read a short text. After this step, no further user interaction is required either for the training or for the recognition phase.

Unlike the client-server approach, in the standalone architecture the models are created, updated and stored on the mobile device. Since there is no communication required with a remote server, the models are updated as soon as the new voice samples are recorded. This also allows the recognition to be instant, without requiring further interaction by the user.

In Section 4.1 it was mentioned that the recording lasts for 60 seconds if it was triggered while the application was in the capturing state or, in case it was initiated by a phone call until the phone call is terminated. It was also mentioned that the recordings are cut into chunks with a maximum duration of 60 seconds each. This

means that during a phone call, multiple voice samples might be recorded. In this case, the system waits until the recording ends and it then converts the new voice samples into voiceprints and begins the recognition process. During this process, the system attempts to recognise the speaker of each voiceprint and if it succeeds, it also updates the corresponding user model.

4.2.3 Architecture comparison

This section describes the main advantages and disadvantages of the two architectures. A performance analysis of both architectures is presented in Chapter 5.

Although the standalone approach allows for instant speaker recognition, the absence of a server introduces a great disadvantage compared to the client-server architecture. This is the inability to provide a cloud storage model and allow cross-device speaker recognition. Unlike the standalone approach where the user models are only available locally on each device, in a client-server approach the models are available to any mobile device running the application as long as the users are authorised to use these particular devices. This allows a user to be recognised by multiple devices, using the same user model.

The main advantage of the standalone architecture is that it reduces the risk of having sensitive information, such as voiceprints, leaked from a remote database or eavesdropped during a client-server communication. Moreover, the standalone approach scales better as the number of registered users increases, although scalability issues at the client-server approach can be eliminated by adopting a cloud computing architecture [44].

Finally, in both approaches the conversion of the voice samples into voiceprints takes place on the mobile device. This requires that the audio file which contains the recording be loaded on the device's heap memory in order for the feature extraction process to begin. Splitting the recordings into chunks with maximum duration of 60 seconds each, given the selected audio format settings described in Section 4.1, the size of each file will be approximately 5.2 MB. This may not cause any problems on most modern Android devices but it does not guarantee that the application will not run out of memory on devices with low capacities. It is worth mentioning that as of 2014 nearly 19,000 distinct Android devices have been reported by OpenSignal's statistics results [45].

Chapter 5

Analysis

This chapter presents the evaluation results of the proposed model. Initially, the accuracy of the Recognito framework is evaluated. In addition, the performance of the two different architectures, described in Chapter 4, is also evaluated with regards to the computation delay of the training and recognition phases.

5.1 Performance evaluation

Recognito's accuracy is evaluated in terms of the four metrics described in Section 2.8: correct match rate, correct non-match rate, false match rate and false non-match rate. The voices of 10 speakers were recorded. As summarised in Table 5.1, 15 different voice samples were recorded per user where 1 is used to create the corresponding user model, 10 are used to update the model and the remaining 4 are used for recognition. For the scope of this thesis, these 10 speakers are referred to as *real users*. The aim is to observe how the performance is affected as the user models are updated.

Table 5.1: Voice samples per user

Model creation samples	1
Training samples	10
Recognition samples with noise	2
Recognition samples without noise	2
Total number of samples	15

In addition, as mentioned in Section 2.8.2.3, the accuracy of Recognito depends on its Universal Model. The more the different voiceprints used for generating the Universal Model, the more accurate the performance of Recognito. In total, 4 different scenarios are presented: the Universal Model is generated from the voiceprints of 10, 20, 50 and 100 users respectively. Voice samples recorded during public speeches at Technology, Entertainment, Design (TED) conferences [46], were used to create a number of *dummy users*.

For each scenario, consider N total number users in order that:

```
N = R + D
```

where R is the number of real users and D the number of dummy users. N represents the number of different speakers used to generate the Universal Model for each scenario and *not* the number of the user models that are created. The number of the user models created is R. The voices of D dummy users are only used to generate the Universal Models (together with the real users). There are no user models created for the dummy users, thus these users cannot be updated nor recognised. Table 5.2 summarises how many users are used to populate the Universal Model in each scenario.

Table 5.2: Number of users per scenario

Scenario	***N***	***R***	***D***
1	10	10	0
2	20	10	10
3	50	10	40
4	100	10	90

Finally, an additional 10 speakers that are referred to as *fake users*, have been recorded. For each fake user, 2 voice samples were recorded in order to test the system's performance in terms of correct non-match rate. The role of these users is to act as imposters and attempt to be recognised by the system when there are no stored user models for them.

All voice samples used in the conducted experiments have the same format as summarised in Table 4.1. The duration of the voice samples of the real, as well as the fake users, is 60 seconds. For the dummy users there are two versions of

voice samples. In the first version the duration of the samples is 60 seconds, while in the second the duration is 10 minutes. The idea is to observe if using dummy users with voice samples of 10 minutes rather than 60 seconds, will create better Universal Models and improve the system's accuracy.

The voice samples used for creating and training each user model were recorded in an enterprise open plan office of approximately 70 employees with low background noise. These samples, as well as the samples of the dummy users, are common for all experiments. The experiments only differ with regards to the voice samples used for the recognition phase and can be grouped into three categories:

Performance without background noise
: The recognition voice samples belong to real users and were recorded in the same environment as the samples that are used for creating and training the models. That is, an open plan office of approximately 70 employees.

Performance with background noise
: The recognition voice samples belong to real users and were recorded in an environment with noticeable background noise. In particular, they were recorded at Stockholm's central subway station during peak hours.

Performance against imposters
: The recognition voice samples belong to fake users and were recorded in the same environment as the samples that are used for creating and training the models. That is, an open plan office of approximately 70 employees.

For each of the above three categories there are two alternatives. In the first alternative, the voice samples of the dummy users have a duration of 60 seconds, while in the second alternative the duration of these samples is 10 minutes.

The aim of the experiments is to find the state where the overall accuracy approaches its maximum and cannot be significantly improved anymore. This is mainly affected by two parameters: the Universal Model and the amount of training needed for a user model.

5.1.1 Performance without background noise

In this experiment, the voice samples used for the recognition phase belong to real users and were recorded in the same environment as the samples that are used for creating and training the models. In particular, they were recorded in an enterprise

open plan office of approximately 70 employees with fairly low background noise.

Initially 10 user models are created. For each model there are 10 *training rounds*, where in each round the model is updated with a new voice sample. We attempt to recognise the speaker of each model after the model is created, as well as at the end of each training round, expecting that the accuracy increases as the model is updated. An important characteristic of this experiment is that while the model of a *User x* is updated round by round, the remaining 9 models stay untrained. This applies for all 10 user models. The results presented below are the average value of a total of 20 recognition attempts, using 2 different voice samples for each user.

Figure 5.1 illustrates how the accuracy increases for each scenario when the duration of the dummy users' samples is 60 seconds. The maximum correct match rate that is reached is 70% and is first reached by scenarios 3 and 4 after the 4th training round. Scenarios 1 and 2 also reach 70% after the 6th and 7th round, respectively. All scenarios are listed in Table 5.2. Figure 5.2 shows how the peak of the correct match rate density changes for each scenario.

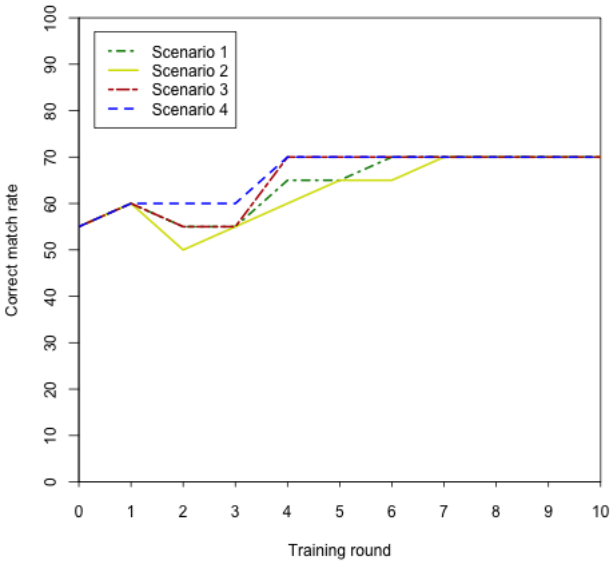

Figure 5.1: Correct match rate for alternative 1

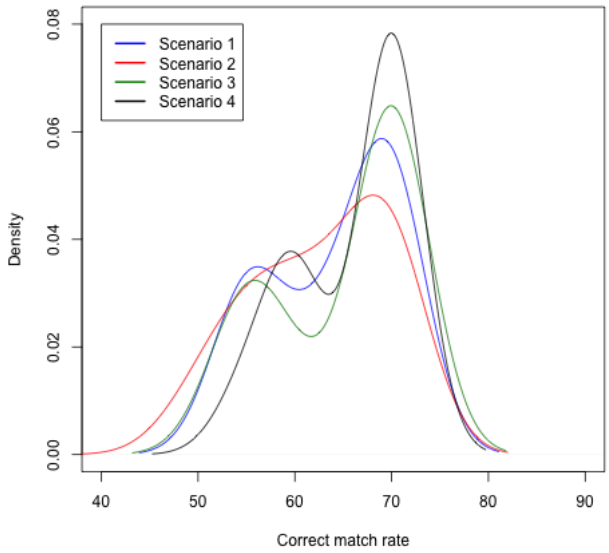

Figure 5.2: Correct match rate density for alternative 1

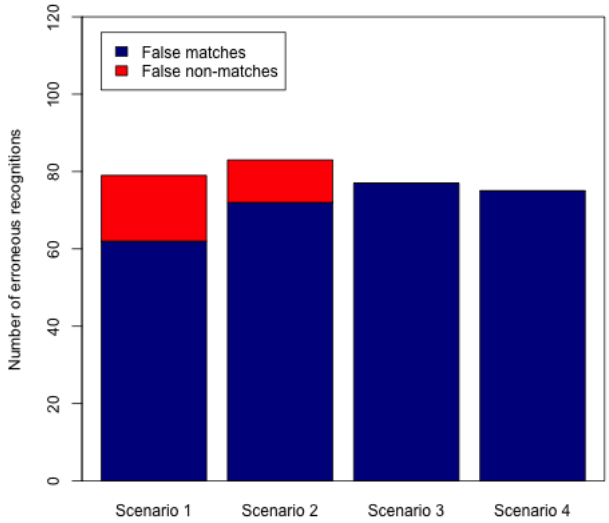

Figure 5.3: Total number of erroneous recognitions for alternative 1

It is worth mentioning that 9% of the total recognition attempts that failed (from all 4 scenarios combined) were false non-matches, while the remaining 91% were false matches. This means that the majority of the failures occurred because the system falsely recognised another existing user instead of the actual one. Figure 5.3 shows the number of errors after a total of 220 recognitions per scenario. The number 220 is derived as follows: 20 recognitions per training round (2 recognitions per user) in addition to 20 recognitions before the first training round started.

Figure 5.4 presents the results of the second alternative, where the duration of the dummy samples is 10 minutes. The longer samples result in a better Universal Model which slightly improves the overall accuracy. For the real users, the same samples and in the same order as in the first alternative, were used for creating and updating the models, as well as for recognising the speakers. The maximum correct match rate achieved in this case is 75%. It is first reached by scenario 4 after the 5th training round and is followed by scenarios 2 and 3 at the end of the 6th training round. Scenario 1 has the same values as in the first alternative, since there are no dummy users involved and its Universal Model was generated only by voice samples of real users. Figure 5.5 shows how the peak of the correct match rate density changes for each scenario.

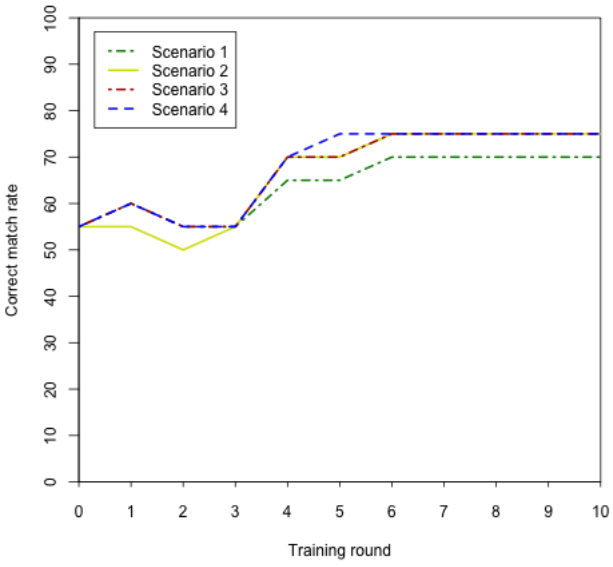

Figure 5.4: Correct match rate for alternative 2

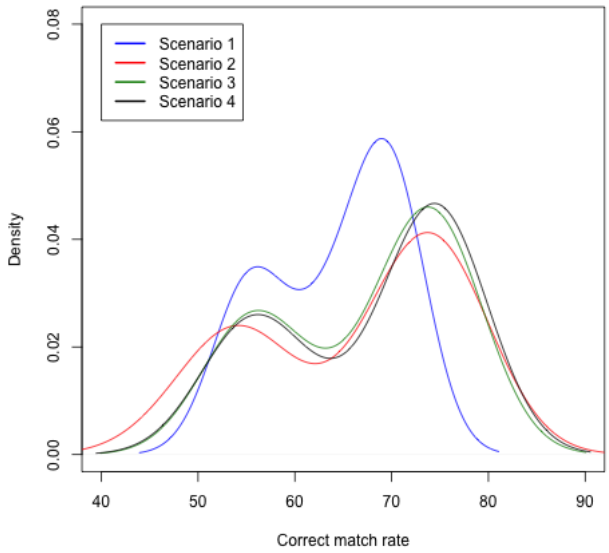

Figure 5.5: Correct match rate density for alternative 2

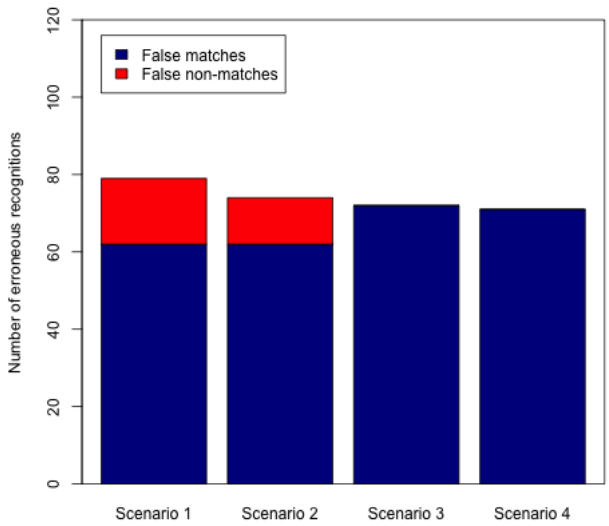

Figure 5.6: Total number of erroneous recognitions for alternative 2

Similar to the first alternative, the majority of all the failed recognition attempts occurred due to false matches. In particular, 90% of all these attempts were false matches, while only the remaining 10% were false non-matches. Figure 5.6 shows the number of errors after a total of 220 recognitions per scenario.

5.1.2 Performance with background noise

In the experiment presented in this section, the voice samples used for the recognition phase belong to real users. However, unlike the experiment presented in Section 5.1.1 these voice samples were not recorded in the same environment as the voice samples used for creating and training the models. In particular, they were recorded at Stockholm's central subway station between 12:00 and 15:00 on a Saturday afternoon and contain noticeable but not excessive background noise.

The process of creating and updating the user models, as well as recognising the users is the same as described in Section 5.1.1. There are 10 training rounds for each model and only one model is updated at any time, while the remaining 9 models stay untrained. There exist two alternatives of this experiment: one where the duration of the dummy users' voice samples is 60 seconds and a second alternative where the duration of these samples is 10 minutes.

The results have shown that all 4 scenarios reached 100% of false-non match rate, regardless of the number of training rounds that preceded the recognition phase. The result was the same for both alternatives since using longer voice samples for the dummy users (second alternative) did not improve the performance.

5.1.3 Performance against imposters

The experiment presented in this section differs from the ones described in Section 5.1.1 and Section 5.1.2 in two ways. First, the voice samples that are used for recognition, belong to fake users. That is, they belong to users that are not stored in the system. Moreover, all existing user models are trained simultaneously. This means that every time a fake user is recognised, all stored models have gone through the same number of training rounds. The recognition phase happens once after the actual user models are created and at the end of each training round. The voice samples used for the recognition phase were recorded in the same open plan office as the one where the samples for creating and training the models were also recorded.

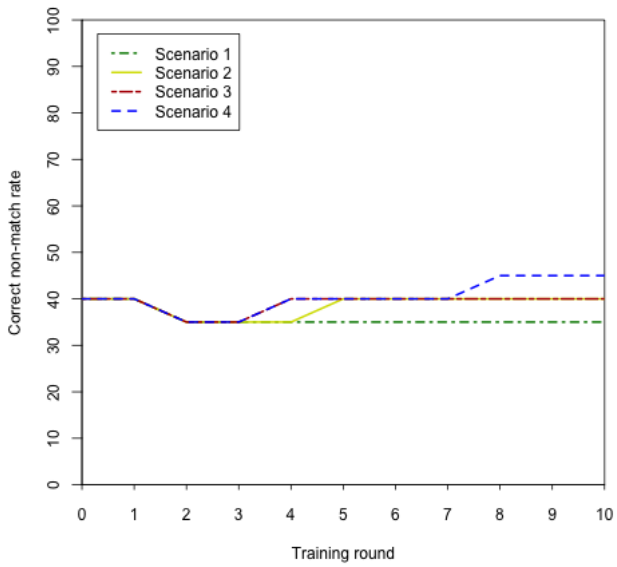

Figure 5.7: Correct non-match for alternative 1

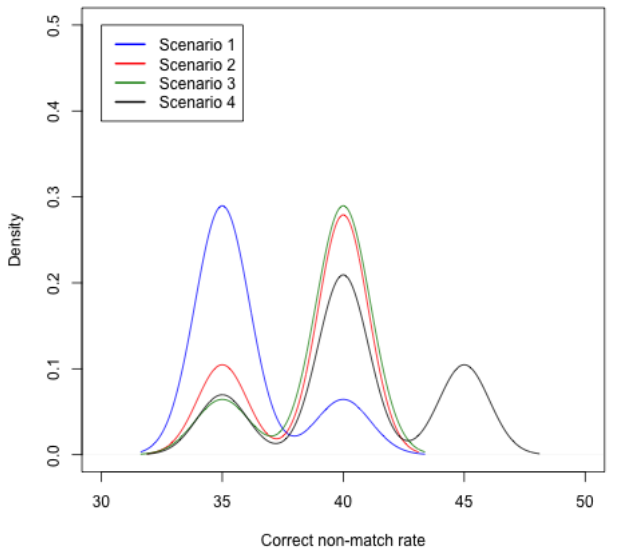

Figure 5.8: Correct non-match rate density for alternative 1

Similar to the experiments presented in the previous sections, there exist two alternatives. Figure 5.7 presents the resulted correct non-match rate after each training round, for the first alternative. None of the scenarios reaches a high correct non-match rate, with the peak being 45% for scenario 4 after the 8th training round. Figure 5.8 shows the correct-non match rate density for each scenario when the duration of the dummy users' voice samples is 60 seconds.

Figure 5.9 illustrates the results of the second alternative. Scenario 1 has the same outcome as in the first alternative since there are no dummy users affecting the Universal Model. For the remaining scenarios, even though longer voice samples were used to populate the Universal Model, there is no significant improvement in performance. The highest correct non-match rate with a value of 45% is reached both in scenario 3 and scenario 4 after the 8th and 6th round, respectively. Figure 5.10 shows the correct-non match rate density for each scenario when the duration of the dummy users' voice samples is 10 minutes.

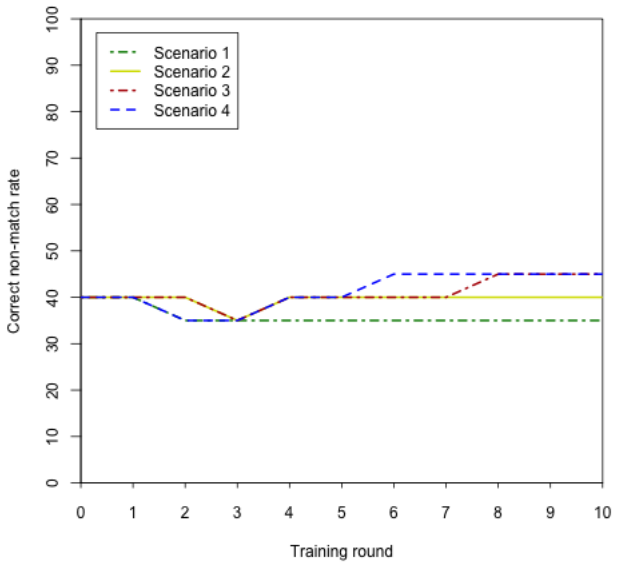

Figure 5.9: Correct non-match rate for alternative 2

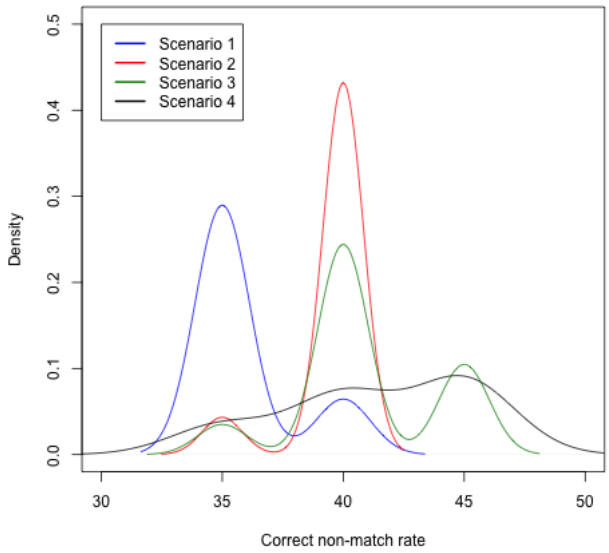

Figure 5.10: Correct non-match rate density for alternative 2

5.2 Evaluation of specific usage scenarios

The experiments presented in Section 5.1 showed that generating a Universal Model from the voice samples of 100 dummy users with a duration of 10 minutes, slightly improves the overall accuracy. In this section, the following three usage scenarios are evaluated when using the mentioned Universal Model:

1. A single user device.
2. A device shared by a couple.
3. A device shared by a 4-member family.

In the first scenario, we assume that a single user is stored in the system, while in the second and third scenario there are 2 and 4 stored users, respectively. For all three scenarios, the voice samples of the real users have been used, as described in Section 5.1 and summarised in Table 5.1. For the second scenario, 10 couples of one man and one woman were formed by unique combinations of the 10 real users. Accordingly, for the third scenario 10 families of two women and two men were formed by unique combinations of the 10 real users. In both cases, the age of the members was also taken into consideration in order to simulate realistic scenarios

of couples and families. Since there are not enough user samples, some of the real users were selected multiple times to form a couple or a family. However, each couple and family is unique with regards to the total members that it consists of.

Similar to the previous section, the experiments can be grouped into three categories:

- Performance without background noise.
- Performance with background noise.
- Performance against imposters.

5.2.1 Performance without background noise

The voice samples used for the recognition phase in this experiment belong to real users and were recorded in the same environment as the samples used for creating and training the models. In particular, they were recorded in an enterprise open plan office with fairly low background noise. The process of creating and updating the user models, as well as recognising the users is the same as in the experiment presented in Section 5.1.1. There are 10 training rounds for each model. In the usage scenarios 2 and 3 only one model is updated at any time while the remaining models stay untrained. This does not apply to the usage scenario 1 since in this case there is always one stored user.

In the first usage scenario, the result is the average value of 20 recognition attempts, 2 for each user. In the second usage scenario the result is the average value of 40 recognition attempts, 2 for each member of a couple multiplied by 10 couples. Accordingly, in the third scenario the result is the average value of 80 recognition attempts, 2 for each member of a family multiplied by 10 families.

Figure 5.11 illustrates how the accuracy increases for each usage scenario as the stored models are trained. The first two scenarios perform slightly better than the third because of the number of users that are stored in the system. Fewer stored users result in lower false match rate. In particular, as shown in Figure 5.12 the false match rate in the single user scenario is 0 since there is always one stored user. In the second scenario, the false match rate is close to 5% of the total errors, while in the third scenario where there are 4 stored users, the false match rate reaches 20% of the total errors.

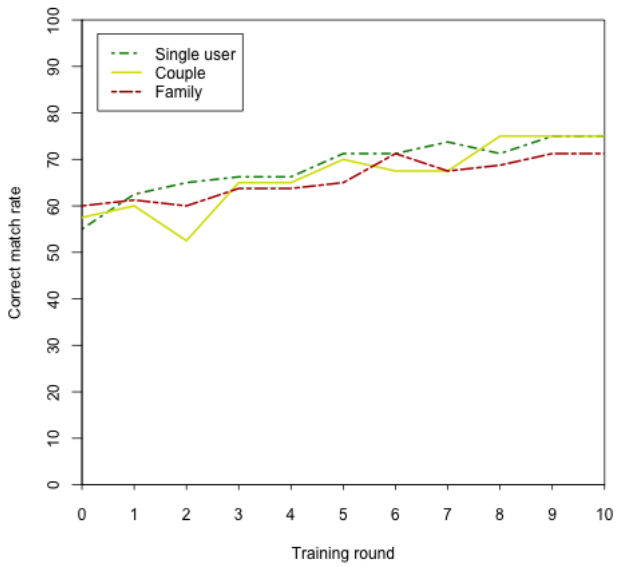

Figure 5.11: Correct match rate of specific usage scenarios

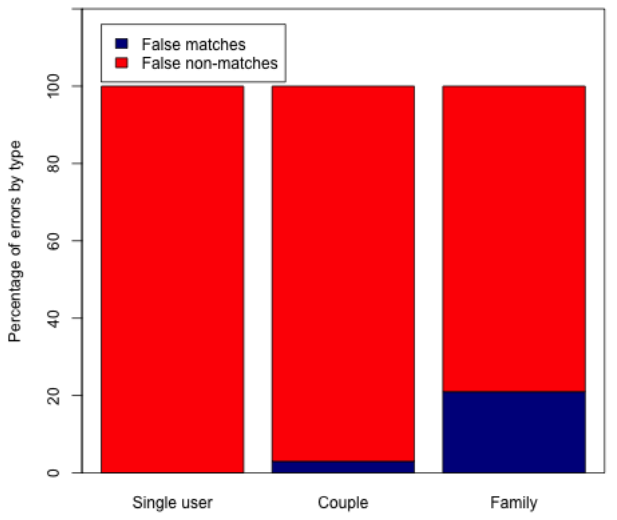

Figure 5.12: Percentage of errors by type

5.2.2 Performance with background noise

Similar to the experiment presented in Section 5.1.2, the voice samples used for the recognition phase in this experiment belong to real users and were recorded in a different environment than the voice samples used for creating and training the models. In particular, they were recorded at Stockholm's central subway station between 12:00 and 15:00 on a Saturday. In addition, the process of creating and updating the user models, as well as recognising the users remains also the same.

The results have shown that all three usage scenarios reached 100% of false-non match rate, regardless of how many training rounds preceded the recognition phase. This outcome is expected given that the recognition also failed for all the users in the experiment presented in Section 5.1.2.

5.2.3 Performance against imposters

The experiment presented here is similar to the one presented in Section 5.1.3. The voice samples used for the recognition phase belong to fake users. In addition, in the usage scenarios 2 and 3 all the existing user models are trained simultaneously. This means that every time a user is recognised, all stored models have gone through the same number of training rounds. This does not apply to the usage scenario 1 since in this case there is always one stored user in the system.

Figure 5.13 illustrates how the correct non-match rate increases for each scenario as the models are trained. Another important outcome is that the correct non-match rate decreases significantly as the number of stored users increases from 1 to 2 and then to 4. The more the stored users, the higher the possibility for a false match.

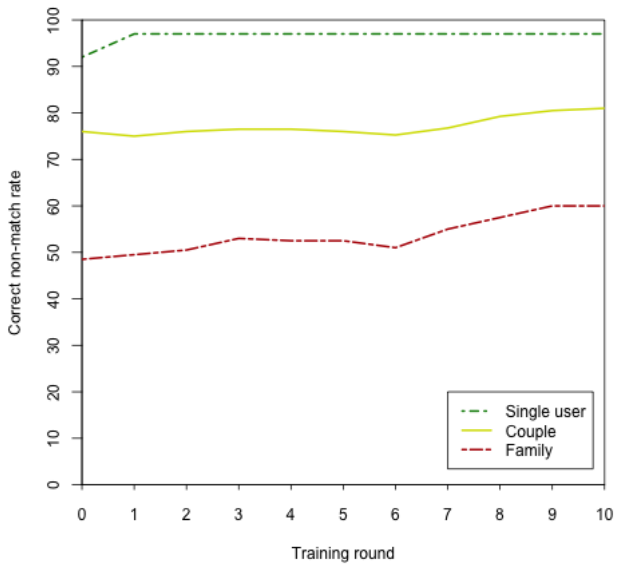

Figure 5.13: Correct non-match rate of specific usage scenarios

5.3 Evaluation of the proposed architectures

This section presents the analysis results of the performance comparison of the two proposed architectures. Their performance is compared in terms of computation delay during the training and recognition phase.

A server was set up and deployed on a MacBook Pro laptop while the Android applications were tested on a Nexus 5 smartphone. Table 5.3 provides additional information about the technical specifications of the two devices.

Table 5.3: Technical specifications

Device	**OS version**	**RAM**	**Processor**
MacBook Pro	OS X 10.10.1	16 Gb	2 GHz Intel Core i7
Nexus 5	Android 4.4.4	2 Gb	2.3 GHz Krait 400

As mentioned in Chapter 4, in both architectures the voice samples are converted into voiceprints on the mobile device. Thus, the delay for creating the

voiceprints is not taken into consideration. The two architectures, standalone and client-server, are compared with regards to the time it takes to update an existing user model, as well as the time it takes to recognise a speaker.

Ten user models are created and stored in the system. Figure 5.14 illustrates the average computation delay of 100 model training operations, as well as the average delay of 100 recognition operations for each architecture. While in the client-server architecture the average delay of the training phase is close to 4 milliseconds, in the standalone architecture the average delay is close to 26 milliseconds. The difference in the recognition phase's computation delay between the two architectures is smaller: the average value in the client-server architecture is close to 5 milliseconds while for the standalone it is approximately 20 milliseconds.

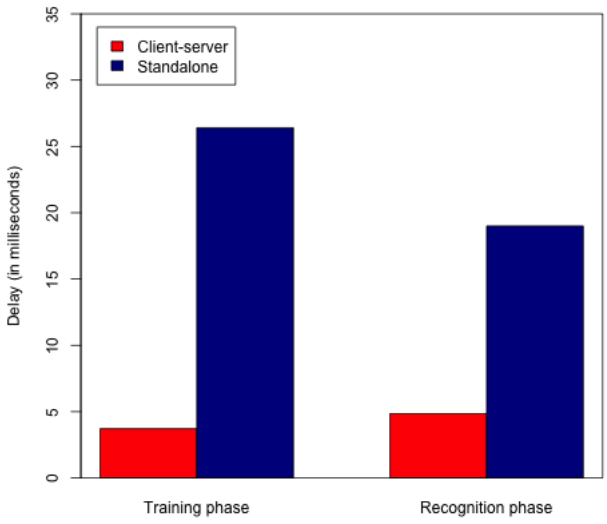

Figure 5.14: Computation delay comparison

Chapter 6

Conclusions and future work

This section describes the challenges that were faced during the implementation of this thesis. In addition, it discusses the results of the experiments, as well as possible improvements in future work. Finally, it brings up reflections on ethical and social risks.

6.1 Challenges

The first challenge that was faced during the implementation process, regards a known bug in one of the Android Application Programming Interfaces (APIs). The Android platform provides an API that performs the required tasks for the Google Voice Search [47] feature, namely *SpeechRecognizer*. This service performs, among others, voice detection, voice capturing and speech recognition. One of its features is giving feedback about whether or not there has been human voice detected. This is used by our implementation in order to make a decision about whether the recording of voice samples should start. In particular, the SpeechRecognizer provides the following callbacks [48]:

- onBeginningOfSpeech()
- onEndOfSpeech()

Once the speech has started, the onBeginningOfSpeech() is triggered 10 seconds after the speaker has paused his/her speech. This 10 seconds duration is the default value and according to the official documentation [49], developers can parameterise this based on their needs. However, there is a bug in Android version 4.4.4, as a result of which the SpeechRecognizer ignores any changes to this value [50]. This bug affects our implementation since it does not allow us to continue recording when a speaker pauses for more than 10 seconds. The

workaround that was used in our project, is that rather than stopping the recording when the onEndOfSpeech() is triggered, we start a 30 seconds timer in the onBeginningOfSpeech() and when this timer has expired, the recording stops.

Finally, there were found some compatibility issues between Recognito and Android. Recognito's implementation relies on the javax.sound.sampled package for capturing, processing, and playback of sampled audio data. This javax API is not supported by Android, making Recognito incompatible with Android. The reason is that Android does not use a regular Java Virtual Machine (JVM) but its own one, named *Dalvik* [51]. Dalvik works with a specific set of Java packages and javax.sound.sampled is not one of them [52]. In order to work around this problem, we replaced Recognito's javax dependencies with a Java class, called *WavReader*, which provides the same functionality without the need of any javax packages. The WavReader class was provided by the Android User Authentication Framework project [20] and it was sightly modified to fit the needs of this thesis. The class can be found in Appendix A.1.

6.2 Conclusions

An analysis of Recognito showed that it is *not* capable of achieving high accuracy results. When the voice samples used for the recognition phase do not contain significant background noise, Recognito reached a moderate 75% accuracy. However, it failed to recognise *any* of the speakers when the voice samples were recorded in a noisy environment. In addition, Recognito performs poorly against imposters as the number of stored users increases. Although, it achieved a 97% correct non-match rate in the scenario where there was only one stored user, this percentage dropped to 60% and 45% when the number of stored users increased to 4 and 10 users, respectively.

The overall performance is slightly improved when we use voice samples of 100 dummy users with a duration of 10 minutes to generate Recognito's Universal Model. As mentioned in Section 2.8.2, the Universal Model is the mean value of a set of voiceprints and, thus is a voiceprint itself. In order to achieve high accuracy even when there is only one stored user, the system (both in standalone and client-server) is initialised with a pre-generated Universal Model of 100 unique voice samples, with a duration of 10 minutes each.

The comparison between the two architectures has shown that there is no significant performance degradation in the standalone version compared to the client-server. Their major difference in terms of computation delay occurs during

the training phase where the standalone architecture has an average delay of 26 milliseconds while the client-server has a delay of approximately 4 milliseconds. Although the standalone's training phase is more than 6 times slower, the difference of 22 milliseconds is not considered to be significant. According to the official Android guidelines, the threshold beyond which users can generally perceive slowness in an application is between 100 and 200 milliseconds [53].

Finally, in order to keep the size of the voice samples under control, all the recordings are split into chunks with maximum duration of 60 seconds each. This, along with the specified audio format settings, ensures that file size of each voice sample will not exceed 5.2 MB. Although loading a 5.2 MB file in the heap memory of a state-of-the-art smartphone will most likely not cause any problems, it does not guarantee that the standalone version will not run out of memory on devices with low capacities.

6.3 Future work

The first step of our future work is to evaluate other speaker recognition systems in order to find one that achieves significantly improved accuracy compared to Recognito. Both the standalone and the client-server architecture use a wrapper around the selected recognition system (in this case, Recognito) in order to make it easily replaceable.

In addition, it would be possible to further improve the accuracy of our system, by combining speaker recognition with other features, such as:

Face recognition
: Use the device's camera to perform face recognition and combine it with speaker identification to improve the likelihood ratio.

GPS
: Make use of the user's current location to increase the likelihood ratio. For example, if a user's speech was captured while he/she was at a location near his/her home, then it is more likely that this speech belongs to this specific user.

Calendars
: Combine the use of GPS with calendars. Many users store in their electronic calendars their daily events. These events usually provide information about the location. If, for example, a user has an appointment at a specific address and the GPS confirms that the user is indeed close to this location, then a

captured voice sample from this location is more likely to belong to this user.

Android wear
Android watches gain popularity rapidly. Each watch is paired and communicates with an Android smartphone. These devices have hardware components such as accelerometers, gyroscopes, GPS and microphones [54]. This allows us to gather additional user information that combined with the information gathered by the user's smartphone can improve the system's overall accuracy.

6.4 Required reflections

Storing biometric information on a server is a controversial approach. Although, as of today, voice cannot be used to uniquely and safely identify a person, it is possible that technology will be capable of achieving this in the future. In such a case, a potential compromise of the server's database would expose sensitive information about all the registered users.

Finally, another issue is the legal frame of storing biometrics. Addressing this, becomes complicated considering the different European Union (EU) and nation-specific legislations, as well as the cultural background behind any written law [55]. Similarly to all kind of biometrics, voice identification is prone to the attacks that Ratha et al. pointed out [56], which raises a series of problems that need to be addressed before considering any voice based identification system as reliable.

Bibliography

[1] Dave Coustan, "How smartphones work," Mar. 2014. [Online]. Available: http://electronics.howstuffworks.com/smartphone.htm

[2] Liane Cassavoy, "What makes a phone a smartphone?" Mar. 2014. [Online]. Available: http://cellphones.about.com/od/smartphonebasics/a/what_is_smart.htm

[3] Carlos Domíngez Sánchez, "Speaker recognition in a handheld computer," Master's thesis, KTH Royal Institute of Technology, 2010. [Online]. Available: http://www.yumpu.com/en/document/view/3230127/speaker-recognition-in-a-handheld-computer-skolan-for-

[4] Joseph P. Campbell Jr., "Speaker recognition: a tutorial," *Proceedings of the IEEE*, vol. 85, no. 9, pp. 1437–1462, Sep. 1997. doi: 10.1109/5.628714

[5] FBI Biometric Center of Excellence, "Speaker recognition," Mar. 2014. [Online]. Available: http://www.fbi.gov/about-us/cjis/fingerprints_biometrics/biometric-center-of-excellence/files/speaker-recognition.pdf

[6] BehavioSec, "Behaviometrics," Mar. 2014. [Online]. Available: http://www.behaviosec.com/wp-content/uploads/2012/01/BehavioSec-Behaviometrics.pdf

[7] Naresh P. Trilok, Sung-Hyuk Cha, and Charles C. Tappert, "Establishing the uniqueness of the human voice for security applications," in *Proceedings of Student/Faculty Research Day*. Pace University, May 2004.

[8] Finjan, "User identification and authentication," Mar. 2014. [Online]. Available: http://199.203.243.203/objects/manuals/9.2.0/User_Identification_and_Authentication.pdf

[9] National Institute of Standards (NIST), "The NIST year 2012 speaker recognition evaluation plan," May 2012. [Online]. Available: http://www.nist.gov/itl/iad/mig/upload/NIST_SRE12_evalplan-v17-r1.pdf

[10] Tomi Kinnunen and Haizhou Li, "An overview of text-independent speaker recognition: From features to supervectors," *Speech Communication*, vol. 52, no. 1, pp. 12–40, Jan. 2010. doi: 10.1016/j.specom.2009.08.009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167639309001289

[11] U. Shrawankar and V. M. Thakare, "Techniques for feature extraction in speech recognition system : A comparative study," *arXiv:1305.1145 [cs]*, May 2013, arXiv: 1305.1145. [Online]. Available: http://arxiv.org/abs/1305.1145

[12] Hong Kook Kim, Seung Ho Choi, and Hwang Soo Lee, "On approximating line spectral frequencies to LPC cepstral coefficients," *IEEE Transactions on Speech and Audio Processing*, vol. 8, no. 2, pp. 195–199, Mar. 2000. doi: 10.1109/89.824705

[13] Namrata Dave, "Feature extraction methods LPC, PLP and MFCC in speech recognition," *International Journal For Advance Research in Engineering And Technology(ISSN 2320-6802)*, vol. Volume 1, no. Issue VI, 2013.

[14] Eslam Mansour Mohammed, Mohammed Sharaf Sayed, Abdallaa Mohammed Moselhy, and Abdelaziz Alsayed Abdelnaiem, "LPC and MFCC performance evaluation with artificial neural network for spoken language identification," *International Journal of Signal Processing, Image Processing and Pattern Recognition*, vol. 6, no. 3, Jun. 2013.

[15] V. Tyagi and Christian Wellekens, "On desensitizing the mel-cepstrum to spurious spectral components for robust speech recognition," in *IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005. Proceedings. (ICASSP '05)*, vol. 1, Mar. 2005. doi: 10.1109/ICASSP.2005.1415167 pp. 529–532.

[16] Douglas Reynolds, "Gaussian mixture models," in *Encyclopedia of Biometrics*, Stan Z. Li and Anil Jain, Eds. Springer US, Jan. 2009, pp. 659–663. ISBN 978-0-387-73002-8, 978-0-387-73003-5. [Online]. Available: http://link.springer.com/referenceworkentry/10.1007/978-0-387-73003-5_196

[17] F.K. Soong and A.E. Rosenberg, "On the use of instantaneous and transitional spectral information in speaker recognition," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 36, no. 6, pp. 871–879, Jun. 1988. doi: 10.1109/29.1598

[18] Sadaoki Furui, "Recent advances in speaker recognition," *Pattern Recognition Letters*, vol. 18, no. 9, pp. 859–872, Sep. 1997. doi: 10.1016/S0167-8655(97)00073-1. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167865597000731

[19] Frédéric Bimbot, Jean-François Bonastre, Corinne Fredouille, Guillaume Gravier, Ivan Magrin-Chagnolleau, Sylvain Meignier, Teva Merlin, Javier Ortega-García, Dijana Petrovska-Delacrétaz, and Douglas A. Reynolds, "A tutorial on text-independent speaker verification," *EURASIP Journal on Advances in Signal Processing*, vol. 2004, no. 4, p. 101962, Apr. 2004. doi: 10.1155/S1110865704310024. [Online]. Available: http://asp.eurasipjournals.com/content/2004/4/101962/abstract

[20] "Android user authentication framework - gitorious," Nov. 2014. [Online]. Available: https://gitorious.org/android-user-auth

[21] "Google buys android for its mobile arsenal," Mar. 2014. [Online]. Available: http://www.webcitation.org/5wk7sIvVb

[22] Niels Henze, "Hit it!: An apparatus for upscaling mobile HCI studies," in *CHI '12 Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '12. New York, NY, USA: ACM, 2012. doi: 10.1145/2212776.2212450. ISBN 978-1-4503-1016-1 pp. 1333–1338. [Online]. Available: http://doi.acm.org/10.1145/2212776.2212450

[23] T. Bray, "Identifying app installations | android developers blog," Mar. 2014. [Online]. Available: http://android-developers.blogspot.se/2011/03/identifying-app-installations.html

[24] "Voice extensible markup language (VoiceXML) version 2.0," Mar. 2014. [Online]. Available: http://www.w3.org/TR/2004/REC-voicexml20-20040316/

[25] "Speaker verification," Mar. 2014. [Online]. Available: https://cafe.bevocal.com/docs/verification/index.html?content=overview.html

[26] "W3c "voice browser" working group," Mar. 2014. [Online]. Available: http://www.w3.org/Voice/voice-implementations.html

[27] IBM, "Speaker verification guide," Dec. 2014. [Online]. Available: http://pic.dhe.ibm.com/infocenter/wvsoem/v6r1m1/topic/com.ibm.websphere.wvs.doc/wvs/wvs_sv.pdf

[28] "IBM - WebSphere voice - family," Mar. 2014. [Online]. Available: http://www-01.ibm.com/software/voice/

[29] Amaury Crickx, "Recognito," Jul. 2014. [Online]. Available: https://github.com/amaurycrickx/recognito

[30] Cristian M. Toader, "User authentication for pico: When to unlock a security token," Master's thesis, University of Cambridge, Cambridge, UK, Jun. 2014.

[31] Frank Stajano, "Pico: No more passwords!" in *Security Protocols XIX*, ser. Lecture Notes in Computer Science, Bruce Christianson, Bruno Crispo, James Malcolm, and Frank Stajano, Eds. Springer Berlin Heidelberg, Jan. 2011, no. 7114, pp. 49–81. ISBN 978-3-642-25866-4, 978-3-642-25867-1. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-642-25867-1_6

[32] "ALIZE," Mar. 2014. [Online]. Available: http://mistral.univ-avignon.fr/index_en.html

[33] "JSTK - a native java speech toolkit," Mar. 2014. [Online]. Available: https://code.google.com/p/jstk/

[34] "CMUSphinx wiki," Mar. 2014. [Online]. Available: http://cmusphinx.sourceforge.net/wiki/tutorialsphinx4/doc/Sphinx4-faq.html#speaker_identification

[35] "voiceid - speaker recognition/identification system in python," Mar. 2014. [Online]. Available: http://code.google.com/p/voiceid/

[36] "bob.spear 1.1.3 : Python package index," Aug. 2014. [Online]. Available: https://pypi.python.org/pypi/bob.spear

[37] M.F.R. Chowdhury, S.-A. Selouani, and D. O'Shaughnessy, "Text-independent distributed speaker identification and verification using GMM-UBM speaker models for mobile communications," in *2010 10th International Conference on Information Sciences Signal Processing and their Applications (ISSPA)*, May 2010. doi: 10.1109/ISSPA.2010.5605556 pp. 57–60.

[38] Jin-Yu Li, Bo Liu, Ren-Hua Wang, and Li-Rong Dai, "A complexity reduction of ETSI advanced front-end for DSR," in *IEEE International Conference on Acoustics, Speech, and Signal Processing, 2004. Proceedings. (ICASSP '04)*, vol. 1, May 2004. doi: 10.1109/ICASSP.2004.1325922 pp. I–61–4 vol.1.

[39] Kevin Brunet, Karim Taam, Estelle Cherrier, Ndiaga Faye, and Christophe Rosenberger, "Speaker recognition for mobile user authentication: An android solution," *8ème Conférence sur la Sécurité des Architectures Réseaux et Systèmes d'Information (SAR SSI)*, 2013.

[40] William A. Wulf, "Responsible citizenship in a technological democracy," Jun. 2014. [Online]. Available: http://www.cvaieee.org/html/resp_citizen/responsible_citizenship.html

[41] Henry Petroski, "Reference guide on engineering practice and methods," in *Reference Manual on Scientific Evidence*, 2nd ed. Federal Judicial Center, 2000, pp. 577–624.

[42] "AudioRecord | android developers," Jan. 2015. [Online]. Available: http://developer.android.com/reference/android/media/AudioRecord.html

[43] "MediaRecorder | android developers," Jan. 2015. [Online]. Available: http://developer.android.com/reference/android/media/MediaRecorder.html

[44] Md. Iqbal Hossain and Md. Iqbal Hossain, "Dynamic scaling of a web-based application in a cloud architecture," Master's thesis, KTH Royal Institute of Technology, Stockholm, Sweden, Feb. 2014. [Online]. Available: http://www.diva-portal.org/smash/get/diva2:699823/FULLTEXT02

[45] "Android fragmentation report august 2014," Aug. 2014. [Online]. Available: http://opensignal.com/reports/2014/android-fragmentation/

[46] "TED: Ideas worth spreading," Nov. 2014. [Online]. Available: http://www.ted.com/

[47] "Voice search – inside search – google," Nov. 2014. [Online]. Available: http://www.google.com/insidesearch/features/voicesearch/

[48] "RecognitionListener | android developers," Nov. 2014. [Online]. Available: http://developer.android.com/reference/android/speech/RecognitionListener.html

[49] "RecognizerIntent | android developers," Nov. 2014. [Online]. Available: http://developer.android.com/reference/android/speech/RecognizerIntent.html

[50] "Issue 76130 - android - RecognizerIntent - EXTRA_speech_input_complete_silence_length_millis not working - android open source project - issue tracker - google project

hosting," Nov. 2014. [Online]. Available: https://code.google.com/p/android/issues/detail?id=76130&q=Speech%20RecognizerIntent.EXTRA_SPEECH_INPUT_COMPLETE_SILENCE_LENGTH_MILLIS&colspec=ID%20Type%20Status%20Owner%20Summary%20Stars

[51] "Dalvik bytecode | android developers," Nov. 2014. [Online]. Available: https://source.android.com/devices/tech/dalvik/dalvik-bytecode.html

[52] "Package index | android developers," Nov. 2014. [Online]. Available: http://developer.android.com/reference/packages.html

[53] "Keeping your app responsive | android developers," Jan. 2015. [Online]. Available: http://developer.android.com/training/articles/perf-anr.html

[54] "Android wear," Jan. 2015. [Online]. Available: http://www.android.com/wear/

[55] "The european regulation on biometric passports," Nov. 2014. [Online]. Available: http://www2.law.ed.ac.uk/ahrc/script-ed/vol4-3/hornung.asp

[56] N. K. Ratha, J. H. Connell, and R. M. Bolle, "Enhancing security and privacy in biometrics-based authentication systems," *IBM Syst. J.*, vol. 40, no. 3, pp. 614–634, Mar. 2001. doi: 10.1147/sj.403.0614. [Online]. Available: http://dx.doi.org/10.1147/sj.403.0614

Appendix A

Recognito

A.1 WavReader.java

```
public class WavReader {
        private String filePath;
        private RandomAccessFile raf;
        private int channels;
        private int sampleRate;
        private int byteRate;
        private int frameSize;
        private short resolution;
        private int length;
        private int payloadLength;

        private long currentPos;

        public WavReader(String filePath) {
                this.filePath = filePath;
                init();
        }

        private void init() {
                try {
                        raf = new RandomAccessFile(filePath, "r");

                        // read and file length payload length fields
                        raf.seek(4);
                        length = Integer.reverseBytes(raf.readInt());
                        raf.seek(40);
                        payloadLength = Integer.reverseBytes(raf.readInt());

                        // get other metadata
                        // channel count
```

```
                raf.seek(22);
                channels = Short.reverseBytes(raf.readShort());
                sampleRate = Integer.reverseBytes(raf.readInt());
                byteRate =  Integer.reverseBytes(raf.readInt());
                frameSize = Short.reverseBytes(raf.readShort());
                resolution = Short.reverseBytes(raf.readShort());

                // set at start of data part
                currentPos = 44;
                raf.seek(currentPos);
        } catch (FileNotFoundException e) {
                e.printStackTrace();
        } catch (IOException e) {
                e.printStackTrace();
        }
}

public int read(byte[] buffer) throws IOException {
                return read(buffer, 0, buffer.length);
}

public int read(byte[] buffer, int offset, int count)
        throws IOException {

        int read = raf.read(buffer, offset, count);
        currentPos += read;
        raf.seek(currentPos);
        return read;
}

public void reset() {
        init();
}

public void close() throws IOException {
        raf.close();
}

/**
 * @return the channels
 */
public int getChannels() {
        return channels;
}

/**
 * @return the sampleRate
 */
public int getSampleRate() {
```

```
                return sampleRate;
        }

        /**
         * @return the frameSize
         */
        public int getFrameSize() {
                return frameSize;
        }

        /**
         * @return the frameLength (payloadLength / frameSize)
         */
        public int getFrameLength(){
                return payloadLength / frameSize;
        }

        /**
         * @return the byteRate
         */
        public int getByteRate() {
                return byteRate;
        }

        /**
         * @return the resolution
         */
        public short getResolution() {
                return resolution;
        }

        /**
         * @return the payloadLength
         */
        public int getPayloadLength() {
                return payloadLength;
        }

        /**
         * Wav files are always little-endian (least significant bytes first).
         *
         * @return false
         */
        public boolean isBigEndian(){
                return false;
        }
}
```

TRITA-ICT-EX-2015:01