h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

CASED

Hochschule Darmstadt

- Faculty of Computer Science -

A Context-Aware Mobile Biometric System

A thesis submitted in the fulfillment
of the requirements for the award of the degree

Master of Science (M.Sc.)

by

Heiko Witte, B.Sc.

Advisor:	Prof. Dr. Christoph Busch
Second Advisor:	Dr. Christian Rathgeb

Issue Date: 13.08.2012
Submission Date: 13.02.2013

Declaration

I hereby declare that I am sole author of this thesis and did not use any further contributions of others than in the referenced bibliography.
Where published or unpublished contributions of others are quoted or analogously used this is clearly indicated.
Any drawings and figures in this thesis are either my original work or referenced in the bibliography.
This thesis has not yet been submitted in equal or similar condition to another examination board.

Erklärung

Ich versichere hiermit, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die im Literaturverzeichnis angegebenen Quellen benutzt habe.
Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder noch nicht veröffentlichten Quellen entnommen sind, sind als solche kenntlich gemacht. Die Zeichnungen oder Abbildungen in dieser Arbeit sind von mir selbst erstellt worden oder mit einem entsprechenden Quellennachweis versehen.
Diese Arbeit ist in gleicher oder ähnlicher Form noch bei keiner anderen Prüfungsbehörde eingereicht worden.

Darmstadt, February 11, 2013 ____________________
Heiko Witte

Abstract

The ubiquitous use of smartphones raises the need for stronger protection of these devices. Traditional authentication methods on mobile devices are still knowledge-based, exhibiting well-known drawbacks. In addition, requests for passwords, PINs, or screen lock patterns represent an interruption of the device usage. Unobtrusive biometric authentication techniques, e.g. background speaker or accelerometer-based gait verification, provide a means to improve the user experience. Since these methods only work within distinct environments, further unobtrusive authentication methods are required.

In this thesis the design and evaluation of a context-aware mobile biometric system is presented. An acquired database, which comprises contextual data of 26 different subjects, is employed for subject modeling and classification, and experiments are presented. Obtained results confirm the feasibility of the proposed system and provide hints for further optimizations.

Kurzfassung

Mit der stetig zunehmenden Verwendung von Smartphones steigt der Bedarf an stärkeren Sicherheitslösungen für diese Geräte. Traditionelle wissensbasierte Authentifikationsmethoden, wie PINs oder Musterabfrage, stellen eine Beeinträchtigung in der Benutzung der Geräte dar. Unaufdringliche biometrische Verfahren, wie Hintergrund-Sprecherverifikation oder Beschleunigungssensor-basierte Gangerkennung, sind ein Mittel um den Benutzungskomfort zu erhöhen. Da diese Methoden jedoch lediglich in fest definierten Umgebungen funktionieren, werden zusätzliche Verfahren benötigt.

In der vorliegenden Arbeit wird ein kontext-sensitives mobiles biometrisches System entwickelt und evaluiert. Eine Datensammlung mit 26 Teilnehmern wird durchgeführt und die erhaltenen Kontextinformationen werden zur subjekt-spezifischen Modellierung und Klassifikation verwendet. Die Resultate bestätigen die Umsetzbarkeit des vorgeschlagenen Systems und enthalten Hinweise für weiterführende Optimierungen.

Contents

List of Figures

List of Tables

1. Introduction

With the introduction of the iPhone in November 2007, the smartphone industry developed into a rapidly growing market. Google and 33 other members of the Open Handset Alliance announced the development of the Android operating system in the same month. The first Android smartphone, the HTC Dream, became available in October 2008. It included a GPS module, a three-axis accelerometer, a digital compass as well as a camera. As the trend continued current smartphones are equipped with even more sensors, e.g. ambient light sensors and gyroscopes. These smartphones rely on an internal memory that holds the Android operating system and can be utilised for the installation of apps[1]. In many cases it can be supplemented by inserting an SD-card. Nowadays the lower boundary for the internal memory of the "top tier" smartphones is 8GB. Storage capacity is growing even faster, e.g. caused by the introduction of high resolution displays and the increased file size of app assets.

As a consequence a huge amount of data can be generated by and stored on modern smartphones. People store their contacts, emails, images, journals, music, videos and other personal data on these devices. Business users can store corporate information and connect to company networks via VPN. With the possibility to store and access private or confidential information, a necessity to secure these devices arises. Conventional authentication procedures employ challenge-response methods, with the PIN or Password being the commonly used in operating systems. Knowledge-based authentication has a well-known drawback: passwords are either insecure or hard to remember. People tend to choose simple passwords and PINs, which are easy to guess, that is, the entire system may be infiltrated by applying a brute-force or dictionary attack. Despite these concerns challenge-response procedures pose an inconvenience, that many users avoid by not activating authentication on their devices [BN10]. A second form of authentication requires the user to present a security token, which must be presented to a reader. An example for token-based authentication is the SIM card in mobile phones. This type of authentication is often supplemented with a knowledge-based method.

Biometrics are a well-studied alternative to the former two authentication types.

[1]Short for "application"

While observation of biological biometric characteristics such as fingerprint, iris, ear or face recognition require an interaction with the subject, other characteristics like gait, keystroke-dynamics or other behavioural characteristics enable unobtrusive authentication use cases. Employing such techniques might lower the barrier for users to activate authentication on their devices. These characteristics are frequently regarded as supplemental, as there is no guarantee that a person is walking or typing at the time authentication is requested. A combination of multiple unobtrusive techniques therefore increases the chance to correctly authenticate a subject.

Within the presented thesis emphasis is put on behavioural characteristics of mobile device usage and thus, on the balance between security and user convenience. One convenience aspect is related to false negative decisions of biometric systems. It is inconvenient for users to repeatedly present e.g. their finger or face in order to succeed in authenticating. While tweaking the parameters of the biometric module might reduce false negatives, it might as well increase the number of false positives, which poses a security risk. Thus, the system should optimised to lower these errors. A source of failure apart from biometric characteristics are environmental conditions like ambient light, noise or accelerations. A biometric system should be deployed in a setting with optimal conditions. This represents an unsolved challenge in the case of mobile biometric systems. The contribution of this work is the proposal and development of a context-aware biometric system, evaluating contextual information for subject-specific modelling. Experimental evaluations confirm the soundness of the approach.

1.1. Objectives

Biometric systems inherently have error rates associated with them. While the performance of such a system highly depends on the sophistication of particular algorithms for a given biometric characteristic, the context also plays an important role regarding the overall performance. This is especially true for mobile biometric systems, where the sensor device is constantly used in different environments.

Deploying biometric systems includes controlling certain environmental conditions. For instance, successfully performing speaker recognition requires the acoustic noise in the area of a sensor to be at acceptable levels. Other examples include a dark environment, which can lead to a significant number of false negative errors of a system performing face recognition, or a person driving a car while the system tries to perform gait recognition.

The approach of this thesis is to develop an unobtrusive authentication method, which is based on contextual information. A context-aware biometric system is designed, which utilises contextual data to decide whether authentication is necessary at all, or how many authentication factors shall be required. Other authentication methods can be deployed as modules in the system, and can be activated by the context-aware subsystem. The context-subsystem shall make sensible assumptions about the trustworthiness of the active user, and only require authentication if an unusual context is detected. Users do not want to constantly enter a PIN or password before using their mobile device. While biometrics provide user-friendly and unobtrusive approaches, the question remains whether authentication is necessary at all in a given context. It may not be necessary to protect a device while being at home, meeting friends or staying with the family, while protection shall be active in untrusted environments. Geographical coordinates can be combined with other features like device usage patterns, average acoustic noise level, ambient light level, and battery state to learn if the device is used in a trustworthy context. The derived security level is then used to intelligently require authentication and the number of necessary authentication factors. For this purpose a data collection is performed to gather training data for the classification algorithm.

The modular architecture for the authentication part of the system is based on the international BioAPI ISO/IEC 19784–1 [ISO06a] standard. As the focus of this thesis is put on the context-awareness of the system and the subjects' behavioural characteristics, the design, development, and evaluation of the context-aware subsystem of the architecture represents the major contribution of this work.

1.2. Organisation of Thesis

In chapter 2 the reader is introduced to fundamentals regarding biometrics and biometric systems. The key components of such systems are discussed as well as metrics to measure biometric performance. Common terms and magnitudes are introduced, which are used throughout the thesis.

In chapter 3 a definition for the term "context" is given and the reader is introduced to the field of context-aware systems. Subsequent sections will elaborate on the modelling of context, supporting (architectural) structures for the development of context-aware systems, as well as general information on data acquisition and evaluation.

The design of a context-aware mobile biometric system referred to as *Modular Biometric Authentication Service System (MBASSy)*, is described in chapter 4. This chapter includes a section on previous work in this area, a short introduction to the BioAPI standard, the proposed system architecture, and implementation aspects.

A data collection was carried out as part of the presented thesis. The feature selection, privacy concerns, the software which was developed to assist in collecting the data as well as the collection process, and the results are described in detail in chapter 5.

In chapter 6 the reader is introduced to the concept of the machine learning technique “support vector machine”, which is later applied for the classification of contextual data. Furthermore, advantages and disadvantages of support vector machines are discussed, and information on available implementations is presented.

The process of building subject-specific models is outlined in chapter 7. In addition, related work and research results in the field of implicit or context-based authentication are presented. Furthermore, the classification performance of the system is evaluated, and discussion of obtained results is given.

Final conclusions, findings and future directions are given in chapter 8.

2. Biometrics

A definition for the term "biometrics" is given by the ISO/IEC JTC1 SC37 [ISO12a]:

"Automated recognition of individuals based on their behavioural and biological characteristics"

Biometrics provide user-friendly ways to authenticate individuals. The last decade indicates a trend to incorporate biometrics into a growing number of use cases. In Germany a contemporary example is the inclusion of facial and/or fingerprint references data in identity documents and passports[1]. Another example for the deployment of a biometric system is the EasyPass border control system[2] at the Frankfurt Airport, which performs automatic face recognition on passengers. A large scale biometric system was developed and deployed by the Unique Identification Authority of India (UIDAI)[3], with the goal to provide a way for the verification of identity claims. The enrolment process includes the extraction of iris, fingerprint and face reference templates.

Biometrics provide a means for storing permanent authentication information. Biometrics cannot be forgotten or stolen like e.g. passwords of knowledge-based or smartcards of token-based authentication. While this is a desirable property of biometrics, it also poses a risk to the privacy of individuals. Standards were developed to ensure privacy, by protecting the generated templates with cryptographic methods [ISO11]. This chapter gives a brief overview of some important aspects of biometrics and biometric systems.

2.1. Categorisation

Biometric characteristics can be grouped into two categories: biological and behavioural. The following definitions stem from the biometric vocabulary document [ISO12a]:

[1]`http://bundesdruckerei.de/de/1548-neuer-personalausweis`, last access: 26.01.2013

[2]`https://www.bsi.bund.de/EN/Topics/ElectrIDDocuments/EasyPASS/EasyPASS_node.html`, last access: 26.01.2013

[3]`http://en.wikipedia.org/wiki/Unique_Identification_Authority_of_India`

"Biological and behavioural characteristics are physical properties of body parts, biological and behavioural processes created by the body and combinations of any of these."

Examples for each category can also be found in the aforementioned document:

Biological These characteristics can be directly derived from biological features of the human body. Examples for such characteristics include face topography, iris structure and finger topography.

Behavioural These characteristic can be derived by observing human behaviour. Examples for such characteristics include gait, voice patterns, key-stroke dynamics or handwritten signature dynamics.

The context-aware components of the authentication framework developed in this thesis utilise behavioural characteristics. The derived usage patterns of a device represent a mental model that expresses how and when a user interacts with it. Environmental aspects, which do not represent biometric characteristics, such as ambient light or acoustic noise level are measured as well.

2.2. Biometric Systems

A biometric system consists of several subsystems, which are defined in this section. Before describing the individual components in detail, it is essential to understand the general workflow of such a system. A detailed description of these concepts can be found in the Standing Document 11 (SD11) of the ISO SC37 [ISO10].

Subjects are added to a biometric system with an adequate enrolment function. Each system captures biometric samples of a subject through sensors. The raw sample is then processed in order to extract distinctive features forming the biometric template, which is subsequently stored in a database. Before adding a new template to a database it can be compared to the ones already present in order to avoid multiple enrolments. The identity claim of a subject can now be verified by comparing the similarity of stored features with the extracted features of a sample. The system can be used for verification and identification purposes. These use cases are described in the respective sections.

A general biometric system is depicted in Figure 2.1.

All definitions in the remainder of this chapter are taken from or based on the Standing Document 11 (SD11) of the ISO SC37 WG1 [ISO10], and shortened where necessary to improve readability. This gives the reader the opportunity to quickly familiarise with the standard.

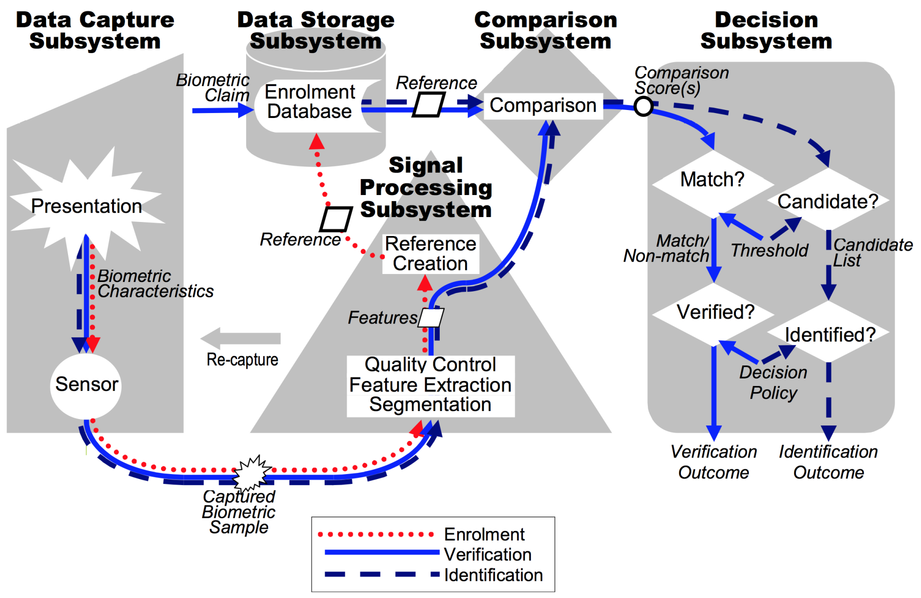


Figure 2.1.: Components of a general biometric system; Source: [ISO10]

Data Capture Subsystem Captures an image or a signal of a subject's biometric characteristics.

Transmission Subsystem Transmits samples, features or references between subsystems. This subsystem is represented by the connections between the other subsystems in Figure 2.1.

Signal Processing Subsystem Processes raw sensor data and performs segmentation, feature extraction and quality control. Creates a biometric template as part of the enrolment process.

Data Storage Subsystem Stores biometric templates. Before being stored in the database, the data can be formatted to conform to a biometric interchange format.

Comparison Subsystem Compares the features extracted from a biometric sample with the features of one or multiple biometric templates. In the verification use case, only a single template is compared with the sample. The results of the comparison are transmitted to the decision subsystem.

Decision Subsystem Uses the comparison score of the comparison subsystem to decide whether an identity claim of a subject is genuine or not. The comparison score must exceed a certain threshold.

Administration Subsystem Controls the implementation, usage and policies of a biometric system.

The following sections provide a brief description of the transactional workflows of a biometric system, namely enrolment, verification and identification.

2.2.1. Enrolment

Subjects are added to a biometric system as part of the enrolment process. A biometric template is created and stored in the database, which is subsequently used in verification or identification processes. The steps involved to enrol a subject are the following:

1. Obtaining a biometric sample from the subject
2. Processing the sample to extract features
3. Quality checks of the derived features
4. Reference creation and conversion to a biometric interchange format
5. Ensure the usability of the generated template by testing verification or identification attempts
6. Repeat process if the initial enrolment is not sufficient

2.2.2. Verification

Verifying the identity claim of a subject involves a (1 : 1) comparison of the acquired sample with the template of the claimed identity. The system can either accept or reject the claim. An identity claim is accepted when the comparison score exceeds a certain threshold, which can be regarded as a minimum confidence level.

The verification process differs from the enrolment process, in that after the quality assurance of the derived features the following steps are performed:

1. Comparison of sample and template
2. Decision based on comparison score

A verification error occurs if a sample of a subject that is enrolled in the system is rejected (false reject error), or in case a claim is accepted but the subject is not enrolled (false accept error).

2.2.3. Identification

A biometric system can be utilised to determine the identity of a subject based on the acquired biometric sample. A $(1 : n)$ comparison of a sample against all or a subset of biometric templates in the database is performed and a candidate list is derived from the scores of this comparison. The score of all candidate identifiers on this list must exceed a certain threshold. Identification is successful, if the subjects' identifier is on the candidate list.

The identification process differs from the verification process, in that multiple comparisons are performed and the results are used to build a candidate list. The identity with the highest score on this list has the greatest similarity to the input sample.

An identification error occurs, when a subjects' identifier erroneously is not on the list (false negative error) or if the subject is not enrolled but its identifier is on the candidate list (false positive error).

2.3. Performance Metrics

The performance of a biometric system can be measured by running verification and identification processes with a test set of users against an enrolment database. The test set may include enrolled users and samples of arbitrary subjects.

The definitions in the following subsections are taken from or based on the biometric performance testing and reporting standard [ISO06b].

2.3.1. False Accept Rate

The false accept rate (FAR) measures the portion of subjects, whose identity claims were erroneously accepted by the biometric system. This error rate cannot be determined in a production system without double-checking the identity of a subject, because there is no way to detect false accepts in the case of automatated verification. However, this would render the automated biometric authentication obsolete. Thus, this error must be determined as part of an explicit evaluation.

2.3.2. False Reject Rate

The false reject rate (FRR) measures the portion of enrolled users, whose identity claim was rejected by the system. Again, this error rate can only be determined by evaluating the reason for the reject in a production system, since the reject might be valid e.g. due to the expiration of an identity document.

2.3.3. Equal Error Rate

The equal error rate (EER) is defined as the setting where FAR and FRR are equal and can be visualised in an detection error-tradeoff (DET) curve. This metric is important for the evaluation of a biometric system, as it represents the lowest achievable total error by balancing convenience (low FRR) and security (low FAR).

2.3.4. Further Metrics

The so far discussed performance metrics are function-level metrics, that are specific to authentication processes. This selection was made as these error rates are the ones a biometric system should be optimised for. This implies that these error rates are acquired at the highest possible level. Errors that occur in deeper levels of a biometric system hierarchy contribute to these error rates.

A set of fundamental performance metrics can be used to measure these lower-level errors, namely the failure to acquire rate (FTA), the false non-match rate (FNMR) and the false match rate (FMR). The FNMR and FMR are algorithmic-level failures, while the FAR and FRR are system-level failures. A false non-match error occurs, when the comparison score of a genuine sample does not exceed the required threshold and accordingly a false match error occurs in the event of a non-genuine sample exceeding the threshold. The FTA can be related to the quality of the feature extraction algorithm, but also to injuries of the subject (e.g. a missing finger) or malfunctioning sensor devices. The failure to enrol rate (FTE) is not considered in this context.

3. Context and Context-Awareness

The Oxford Dictionary of English [SS11] defines ‘context’ as *“the circumstances that form the setting for an event, statement, or idea, and in terms of which it can be fully understood”.* An alternative definition of context in the dictionary is related to natural language: *“the parts of something written or spoken that immediately precede and follow a word or passage and clarify its meaning”.* While these definitions are in line with the tacit understanding of context, these do not fully capture what is understood as context in the field of ubiquitous computing (ubicomp) or human-computer interaction (HCI).

HCI represents a research field which focuses on the user interaction aspects of computing systems. A definition of this field is given by the ACM Special Interest Group on Computer-Human Interaction (SIGCHI): *“Human-computer interaction is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them.”* — [HBC$^+$13]

Improving the usability of computer applications and systems is one of the core goals of HCI. While a major aspect of HCI is the design, implementation and evaluation of user interfaces for the desktop model, a paradigm, that is focused on computing devices which have been embedded into everyday objects and activities, emerged in recent years. The phrase ‘ubiquitous computing’ was first mentioned by Mark Weiser as part of his work at the Xerox Palo Alto Research Center (PARC). He published several papers and articles on this topic [Wei91; Wei93a; Wei93b] and is generally regarded as the “father” of this particular research area.

One aspect of ubicomp is that technology fades into the background. Computing devices shouldn’t be exposed that much, but instead provide their service without requiring user interaction. This is also in line with the idea, that computing devices should fit the user instead of requiring interaction and, thus, forcing him or her into their world. One way to achieve such adaptable computing platforms is to use context information to provide services in the background without requiring user attention or interaction. This goal drove the design and development of the authentication system in the presented thesis.

Visionary examples for ubiquitous computing devices and services include the smart refrigerator[1] that automatically creates shopping lists based on its contents, to the already near ubiquitous location based services, that are the foundation of many thriving businesses.

The remainder of this chapter provides a definition of the term 'context', additional information on the categorisation, and features of context-aware applications. Further, context-modelling techniques are discussed and the reader is introduced to the processes of acquiring, preprocessing, and evaluating context data. This lays the foundation for the design and implementation of the multi-biometric authentication system for smartphones and other portable mobile devices.

3.1. Definition of Context

In order to make informed decisions in the system design phase, a correct understanding of the terms 'context' and 'context-aware' is essential. This subsection provides a brief overview of some attempts to define the terms and closes with a statement regarding which definition is used in this thesis.

Schilit et al. were the first to provide a definition for 'context' and 'context-aware computing applications'. According to them, 'context' is defined by the identity of the user, as well as *"the location of use, the collection of nearby people, hosts, and accessible devices, as well as to changes to such things over time."* [SAW94]. Other factors such as lighting condition, noise level, communication bandwidth, or social situation are mentioned as interesting features of a context description. Schmidt et al. emphasise, that a context is characterised by more information than just the physical location [SBG99].

Dey and Abowd elaborate on various attempts to define context [AD99]. They claim that previous definitions are problematic in that they are based on examples or define 'context' as 'situation' or 'environment', which are synonyms perse. While they also use a synonym, they do not define context as a special type of situation, but instead as the information that can be used to describe a situation:

"Context is any information that can be used to characterise the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves." – [AD99].

[1] `http://en.wikipedia.org/wiki/Internet_refrigerator`

Based on this abstract definition of ‘context’, Dey and Abowd define the term ‘context-aware’ as follows:

“A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user’ s task.” — [AD99].

Contrary to definitions that are centred around the device or environment [Pas98; SAW94], this definition puts the user into the centre of contextual considerations. This definition fits with the goal of this thesis, which is enhancing the usability of authentication systems in order to increase the convenience for users. The task of the user in this case is gaining access to his/her device. Thus, when ‘context’ is mentioned in this thesis, it is in accordance to the definition of Dey and Abowd.

3.2. Categorisation

This section provides an overview of attempts to categorise contextual information and context-aware applications. Taxonomies for contextual information have been suggested by several researchers. Schilit et al. were among the first, who classified the important aspects of context as [SAW94]:

- User identity
- Other people nearby
- Proximate resources

Schmidt et al. proposed a hierarchical feature space, with general human factors and the physical environment as the primary categories, which were then specified in more detail. Examples of human factors are the user, the social environment, and tasks. The physical environment category consists of subcategories for conditions, infrastructure and location. Context history is represented as the time dimension of the feature space. This categorisation is depicted in Figure 3.1.

Dey and Abowd define the intent of context-aware application to monitor the *“[..] who’s, where’s, when’s and what’s [..] of entities”* [AD99]. They derive four primary categories of context data, namely:

- Identity
- Location

- Time
- Activity

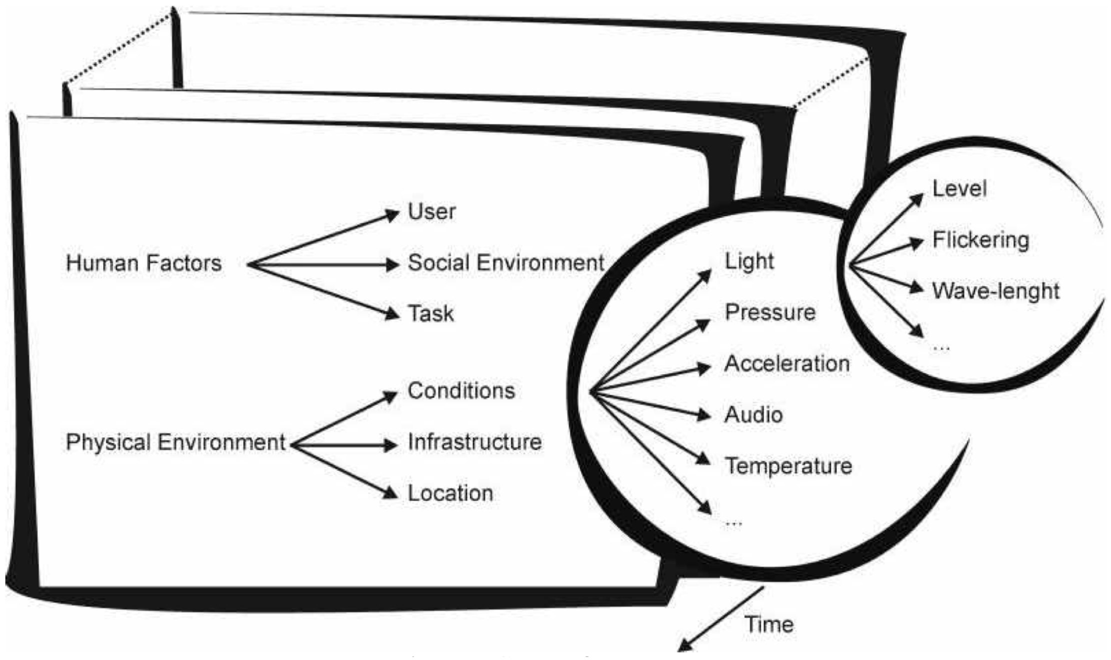


Figure 3.1.: A categorisation of context information; Source: [SBG99]

These categories are used as indices into a secondary feature space and thus, should be supported by every context-aware application. Secondary contextual information directly depends on primary data. If the identity of a user is known, more specific information such as age, address, job, etc. can be obtained. Other context information can be based on the location of a subject. Contemporary examples are location based services, that users can access, e.g. with smartphones or tablets. Activity context information answers the question of what is happening in the current situation. Is the user walking or driving? Is he/she sitting on a couch, watching TV or dancing in a club? This type of contextual information belongs into the activity category.

In this context model it is not defined how the categories relate to or depend on each other. For example, time can depend on the timezone of a subject and thus on the location. It could also be location-independent as in the case of Lamports' logical clock [Lam78] or time intervals in the form of duration constraints. Since the context model in the presented thesis depends on temporal information, every context information type is annotated with a timestamp. The timestamp is based on the system time of a device and is trusted by default. Time is recorded as real time, which depends on the timezone of the user. This clarifies the undefined relation between the context categories for the purpose of the presented thesis. A slightly modified version of Deys' context model is depicted in Figure 3.2.

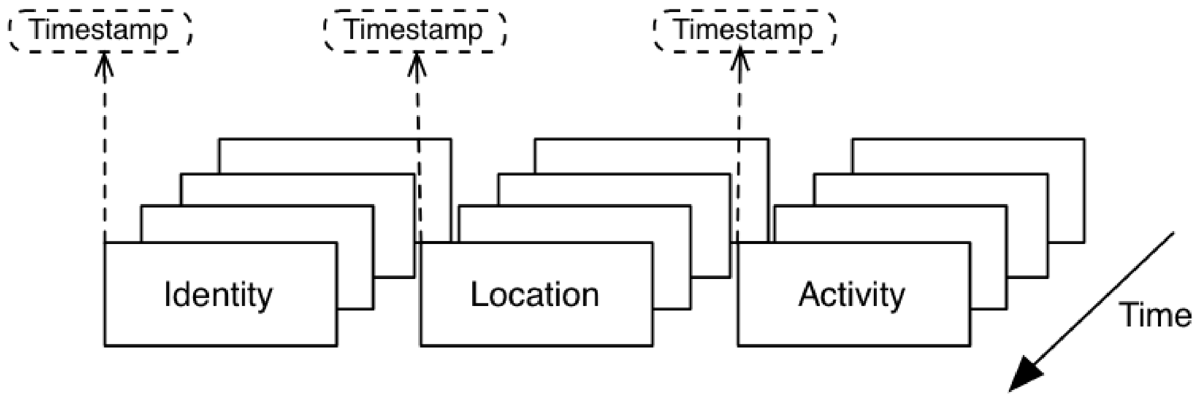


Figure 3.2.: Basic context information categories

Numerous categorisations of context-aware applications have been proposed in literature. The definitions of Schilit [SAW94] and Pascoe [Pas98] are very similar and have been superseded by the proposed categorisation of Dey and Abowd, who describe three distinctive categories [AD99]:

1. Presentation of information and services
2. Automatic execution of services
3. Tagging and storage of context information

These categories can be used to classify the context component of the authentication system in the presented thesis. The classification of the current context to determine the required amount of authentication factors, is a combination of category two and three. Contextual information is utilised in order to infer the strength of the association between the current context and the subject, which in turn is utilised to configure the authentication process.

3.3. Acquisition

Contextual information must be obtained by a background process and, thus, require no user input. As modern smartphones are equipped with a multitude of sensors, a rich set of contextual information can be acquired by applications. The following sections introduce the reader to the type of sensors embedded in the vast majority of modern smartphones, as well as the problem of data granularity or accuracy, and the concept of multi-sensor information fusion.

3.3.1. Sensors

Sensors are the primary source for contextual information. Moschgath identifies three sensor types [Mos13]:

Physical Sensors These sensors provide information about the physical environment. They measure e.g. temperature, acceleration, humidity, geographical position, or light and noise levels

Logical Sensors These sensors provide state information about the computing environment. Examples include battery percentage, number of open network connections, system time, or results from arbitrary database queries.

Interview Components These components are used to acquire information that only the user can provide. This includes information about the user identity or explicit preferences. As the name implies, interview components require user interaction and thus, are not considered in the presented thesis.

While this provides an abstract classification of sensors, other researchers mention specific types. Schmidt et al. focus on physical sensors in their work. They discriminate between three physical sensor categories [SBG99]:

Optical/Vision These sensors provide visual information like light intensity, type of light, colour temperature etc. A camera is regarded as a composition of optical sensors.

Audio Information like frequency, acoustic noise level or loudness can be obtained from raw audio data. A microphone is an example for an audio sensor.

Motion This includes accelerometers that measure acceleration, or gyroscopes that apply the principle of angular momentum to measure orientation.

Location These sensors acquire position information via GPS, cell-site triangulation, or other methods.

Bio-Sensors These sensors measure biological characteristics like pulse, blood pressure or skin resistance.

Specialised Sensors This category includes all sensors that cannot be classified as part of one of the other categories.

Building a context-model for an active user of the authentication system requires the input of a multitude of sensors. Logical sensors are used to obtain time information and usage patterns such as the activation and deactivation of the screen, incoming and outgoing calls including checks if the associated number is part of the contact list and changes to the battery state. Physical sensors are used to determine basic information about physical activity (acceleration), location information, as well as acoustic noise and ambient light levels. The preprocessing of the obtained sensor data is described in section 3.4.

Table 3.1 gives an overview of types of physical sensors, that are embedded in modern smartphones and associates them to the aforementioned categories.

Table 3.1.: Available Sensors in modern smartphones

Sensor Type	Description	Category
Accelerometer	Measures the acceleration on three axes	Motion
Gyroscope	Measures orientation based on angular momentum	Motion
A-GPS	Determines position coordinates	Location
Camera	Can take high resolution images of the environment	Optical/Vision
Microphone	Records audio signals	Audio
Light Sensor	Measures ambient light levels	Optical/Vision
Compass	Measures the magnetic field and provides orientation information	Location
Proximity	Measures the proximity of objects	Various, often Optical/Vision
Temperature	Measures the temperature of hardware components	Specialised Sensor
Humidity	Measures the relative air humidity	Specialised Sensor
Touchscreen	Measures touch input	Specialised Sensor

Not all listed sensors are available in all smartphones and it depends on the operating system if it exposes access to them. Nevertheless, all of the above mentioned sensors can be accessed through the Android API[2].

3.3.2. Granularity

A general problem occurs when dealing with sensor data. Physical sensors are a means to transform continuous analog signals into discrete digital values. While the discretisation inevitably leads to errors, the accuracy of measurement of the analog component also has an impact on the overall quality of the acquired data. Sensor data obtained from heterogenous devices is not guaranteed to meet comparable accuracy levels or data ranges. These facts have to be considered in the design phase of the learning algorithm, since a model trained with fine grained data might not provide an adequate generalisation for coarse grained data, or vice versa.

Since the accuracy of sensors has an impact on the model building, it also has an impact on the evaluation. Devices from different manufacturers may include different sensor models or sensors from different vendors. Thus, a normalisation and transformation of the data would be required if a model shall be evaluated with data acquired on different devices. Raw data should be transformed into suitable ranges as part of the preprocessing step. For instance, instead of directly working with position coordinates, a grid of a specific size could be applied to classify the geographical location data. The grid size could be based on the accuracy of the sensor. Another common preprocessing step is a binning of continuos variables, in order to derive discrete categories.

Another example is the use of acceleration, acoustic noise, magnetic field, and ambient light information. Instead of utilising sequences of raw data, the mean and median of the recordings can be computed and stored for evaluation. While the raw data enables a detailed analysis of features contained in the recorded signals, it also increases the complexity of the model and the evaluation. Developing and applying algorithms for sophisticated feature extraction would make it hard to keep the focus on the development and evaluation of the context-aware system. Thus, a coarser granularity is applied to enable the development and analysis of multiple sensors within the time constraints of the presented thesis.

[2]`http://developer.Android.com/reference/Android/hardware/Sensor.html`

3.4. Preprocessing

An essential aspect of building a context-model represents the definition of the data types and associated ranges of context information. Since a situation shall be characterised by this information, the use of raw data is impractical. Instead, discriminative features should be extracted from the data and stored for reference purposes. Extracting features from raw data is a process, which may involve numerous steps:

Transformation For instance, subtracting a real number from raw accelerometer data to account for the effect of gravity, and centre the data around zero.

Normalisation A special type of transformation. For example the mean normalisation of raw data to obtain comparable ranges across features is referred to as feature scaling.

Mapping For instance, mapping categorial attributes such as the direction of a call to a binary value.

Filtering Removing noise from raw data is an examples for filtering.

Reduction Instead of using the complete set of input data, it is reduced by a function, e.g. computing the mean or median.

None of these steps are mandatory and an application of one of these is determined by the requirements of the model. Adequate methods for the preprocessing of the sensory data used in this work are discussed in chapter 7.

3.5. Multi-Sensor Information Fusion

Combining captured data from various sensors to improve the overall quality of the results is referred to as multi-sensor information fusion [XS02]. A simple example is the combined utilisation of WiFi SSIDs, acoustic noise and GPS to determine whether the subject stays indoors or outdoors. A rule that fuses the results of these sensors may require the device to be logged onto a WiFi network, and an acoustic noise level that does not exceed a certain threshold.

By employing multi-sensor information fusion, the accuracy, robustness, and field of view of a system can be improved. A set of sensors can either be homogenous or heterogenous, which determines the type of fusion. By using a set of homogenous sensors, the overall accuracy of the measurements can be improved by combining the results, whereas using heterogenous sensors increases a systems field of view and may lead to enriched knowledge.

Another aspect of multi-sensor information fusion is the level at which sensor data is fused. Fusing raw data without any preprocessing represents the lowest level and is referred to as data level fusion. Extracting features from raw data and combining those into a single feature vector is referred to as feature level fusion, and combining the results at the end of each individual processing chain is referred to as decision level fusion.

There is no guarantee that a sensor delivers accurate data over time. A good example are the location providers in smartphones, which can be based on GPS or network and WiFi information. It is inherently error prone to derive position information from network or WiFi data, as these deliver only rough estimates. The error increases when a subject moves away from the WiFi access point, so that two location fixes recorded at two different points in time can vary greatly. A location fix is a pair of latitude and longitude at a given time. Even the accuracy of GPS is limited to 10m deviation in the best case. However, the effect of inaccuracies or errors resulting from the subjects' distance to the WiFi access point can be mitigated by using a location database, which is indexed by SSIDs, to retrieve a location estimate, which in turn can be used to speed up the satellite search of the GPS. Other sensors are also subject to errors. To account for these inaccuracies, the data obtained from multiple sensors can be combined to improve the overall result.

While position information is a key feature of the context model in this work, environmental conditions at specific locations may also represent valuable discriminative features. For instance, in case the position information includes an error of several hundred meters, environmental information such as brightness, noise levels or WiFi SSID provide hints about the actual position. For example, if a subject is sitting in an office and the location sensor returns a position that is outside of a building and close to a busy street, the noise level can provide information about the actual setting. The detection performance can be increased further by extending the feature space or by extracting richer features from the raw data. For the previous example, a classification of noise types might deliver more valuable insights regarding the environment.

Another way to fuse multiple sensors is to apply weights to the results as part of a decision process. Weights can be computed for each feature, according to the importance of the feature in the model. This topic is discussed in more detail in chapter 7.

If a sensor fails and does not deliver data, it is marked as a missing value. This can be resolved as part of a preprocessing step, e.g. by replacing missing values with a constant or the mean of the specific feature.

4. Context-Aware System Design

A context-aware mobile biometric system is designed and partially implemented in the presented thesis, which represents the basis for a data collection and analysis. An important aspect of the system is the flexibility to add sensors and group them into categories, as well as to attach preprocessing elements to the sensor groups. A snapshot of the last preprocessed data is held in a context object and archived to a persistent store for later reference.

This chapter provides information on concepts of the Android operating system, the BioAPI standard, previous work, and the context-aware system architecture.

4.1. Android

The Android operating system[1], which is developed by Google and the Open Handset Alliance, had an estimated market share of 68,1%[2] in the second quarter of 2012. An advantage of choosing Android over other platforms for research in the mobile space is, that it has a very permissive API, thus allowing developers to use the features of modern smartphones to a great extent.

Another rational for the platform choice is, that all research carried out by the biometric research group at the Center for Advanced Security Research Darmstadt (CASED)[3] is focused on Android, so building upon this experience is a natural fit.

The following sections briefly describe some of the fundamental concepts of the Android API, that are important to understand.

[1] `http://www.android.com`

[2] `http://www.heise.de/newsticker/meldung/Marktforscher-Ueber-100-Millionen-Androiden-ausgeliefert-1659638.html`

[3] `http://www.cased.de`

4.1.1. Intent

According to the Android developer documentation [Goo13], *"an Intent is an abstract description of an operation to be performed"*[4]. Thus, an *Intent* is used to start an *Activity* or *Service*, which are described in the next subsection. Furthermore, a discrimination between *Implicit Intent* and *Explicit Intent* is required. While the operating system dispatches an *Implicit Intent* to components that match certain criteria, i.e. the action and category specified in an *Intent*, the target component is specified explicitly by its package path in an *Explicit Intent*.

4.1.2. Activity

An *Activity* is the main part of a user interface. It contains interface elements such as buttons, sliders, tables, image views, or *Fragments*[5]. *Fragments* are groups of interface elements that can be reused in other activities and have their own lifecycle management, while still depending on the hosts lifecycle. For instance, when an *Activity* is closed, so are all *Fragments* it contains.

The lifecycle of an *Activity* consists of the following phases:

- `onCreate`
- `onStart`
- `onResume`
- `onPause`
- `onStop`
- `onRestart`
- `onDestroy`

The lifetime of an *Activity* is determined by calls to its `onCreate` and `onDestroy` methods. The other methods are related to the user-visible part of the *Activity* lifetime, such as moving to the back- and foreground.

Since an *Activity* represents an interface to the user, it is only used as part of the data collection application in the presented thesis. The context-aware system is configured in code without any interface elements.

[4]`http://developer.android.com/reference/android/content/Intent.html`
[5]`http://developer.Android.com/guide/components/fragments.html`

4.1.3. Service and WakeLock

A *Service* is an operation that can run in the background, even when a user closes the foreground *Activity* of an application. This feature differentiates Android from iOS[6], since it enables true multitasking for applications (iOS supports multitasking on the operating system level, but exposes only a limited API for developers). It is important to note that a *Service* is neither spawning a separate process, nor is it running in a separate thread. Thus, computation intensive tasks shall create threads of their own when running as part of a *Service*. When a *Service* crashes, it can be automatically restarted by the operating system, so that continuous operation is possible.

The lifecycle of a *Service* is comprised of fewer methods than that of an *Activity*:

- `onCreate`
- `onStartCommand`
- `onDestroy`

A *Service* is paused when the user puts the device to sleep. Furthermore, components of a smartphone such as sensors are deactivated. In order to keep a *Service* and sensors active in such situations, a *WakeLock* can be registered within an application, that keeps the CPU and/or the screen of the device active.

Performing computation intensive tasks in the background leads to a high energy consumption of the application and, thus, to a reduced battery life of the device. Hence, a *WakeLock* shall only be active for a limited period of time. In the next section, the *AlarmManger* is introduced, which enables a scheduled activation of a *Service*.

4.1.4. AlarmManager

The *AlarmManager* provides an API to schedule an *Alarm*. An interesting property of the *AlarmManager* is, that the operating system keeps timers active, and is able to wake up the device when an *Alarm* goes off. This implementation is more energy-efficient compared to permanently enabling a *WakeLock*. In order to schedule an *Alarm*, the time and an interval for the repeated execution, as well as an *Intent* must be specified. When the *Alarm* goes off, the registered *Intent* is broadcasted by the operating system. This enables a scheduled execution of an *Activity* and/or *Service*.

[6]`http://www.apple.com/de/iphone/ios/`

The *AlarmManager* is used in this work to activate the context-aware system in predefined intervals. Thus, the energy consumption of the application is optimised by keeping *WakeLocks* active for a limited time instead of permanently.

4.1.5. Sensors

Smartphones include a multitude of sensors, e.g. an accelerometer, GPS, digital compass, light, proximity, and temperature sensor. The Android API provides a general interface for querying sensors, that can be used in applications. An overview of the supported sensor types is given in table Table 4.1[7]. While the Android API supports a broad range of sensors, it is not guaranteed that every device includes all of these.

Table 4.1.: Sensor types supported by the Android API

Sensor	**Type Specifier**	**Commonly Available**
Accelerometer	TYPE_ACCELEROMETER	Yes
Temperature	TYPE_AMBIENT_TEMPERATURE	No
Gravity	TYPE_GRAVITY	Yes
Gyroscope	TYPE_GYROSCOPE	No
Light	TYPE_LIGHT	Yes
Linear Acceleration	TYPE_LINEAR_ACCELERATION	Yes
Magnetic Field	TYPE_MAGNETIC_FIELD	Yes
Air Pressure	TYPE_PRESSURE	No
Proximity	TYPE_PROXIMITY	Yes
Humidity	TYPE_RELATIVE_HUMIDITY	No
Rotation Vector	TYPE_ROTATION_VECTOR	No

An instance of a specific sensor can be retrieved from the *SensorManager* object that is associated with the application context. In order to receive data from a sensor, a *SensorEventListener* must be registered. The *SensorEventListener* delivers data in the configured intervals by calling the `onSensorChanged` method.

[7]`http://developer.Android.com/reference/Android/hardware/Sensor.html`

4.2. Previous Work

A strong motivator for the development of a context-aware system stems from research on unobtrusive authentication methods at CASED. The Modular Biometric Authentication Service System (MBASSy)[8] was developed [WN10] and used for data collection, testing and demonstration purposes. Authentication methods are implemented as modules which can be loaded into the system, and subsequently used to authenticate a subject if the device is activated in a locked state.

Modules are installed as separate applications. MBASSy exposes its database via a *ContentProvider*. A *ContentProvider* represents a means to perform inter-process communication. Modules authenticate themselves to the system by passing the authentication token, which is retrieved during the registration process, to the *ContentProvider*, so that a module cannot access the data stored by other modules.

The communication interface between the modules and MBASSy is based on *Explicit Intents*, so that malicious applications cannot intercept calls to the MBASSy database.

MBASSy discriminates between background and foreground modules. This discrimination was chosen to determine the primary execution order of modules in the event of an authentication request. Background modules do not require user interaction and, thus, are called prior to foreground modules. In addition to the fixed execution order of module categories, modules within the categories can be prioritised. For instance, if only two foreground modules are installed, the system executes them in the order specified by the user. Depending on the configuration, a single positive authentication result may be sufficient to unlock the system.

Since software interfaces for a biometric authentication system are defined in the BioAPI standard, future implementations shall adhere to it. The presented thesis aims at augmenting the authentication system by adding a context-awareness layer.

The following sections provide an overview of design concepts and architectural decisions, as well as insight into implementation details of the system.

4.3. System Architecture

In this section, an architecture for a context-aware mobile biometric system is presented. Inter-relations of its components, as wells as exemplary configurations are presented. The system is comprised of three key components, which are depicted in Figure 4.1:

[8]`https://www.dasec.h-da.de/research/biometrics/mbassy/`

1. **Context Subsystem**: Within the context subsystem contextual information is collected and processed in order to build a subject-specific model. After deriving the model from training data, a classification process computes the probability of the association between the current context and the genuine subject. This probability, or confidence value, can be used to decide whether authentication is necessary at all, or how many factors shall be required.
2. **Authentication Subsystem**: The authentication subsystem is based on the Java BioAPI standard. Biometric service providers (BSPs) can be registered in the BioAPI Framework and utilised by the context subsystem.
3. **Result Action Subsystem**: Software components can register to receive information on (un-)successful authentication attempts and execute according actions. An interface for the notification on authentication results is implemented in the result action subsystem. An example for such a use case is the combination of biometrics and token-based authentication as presented by Derawi et al. [DWMB12].

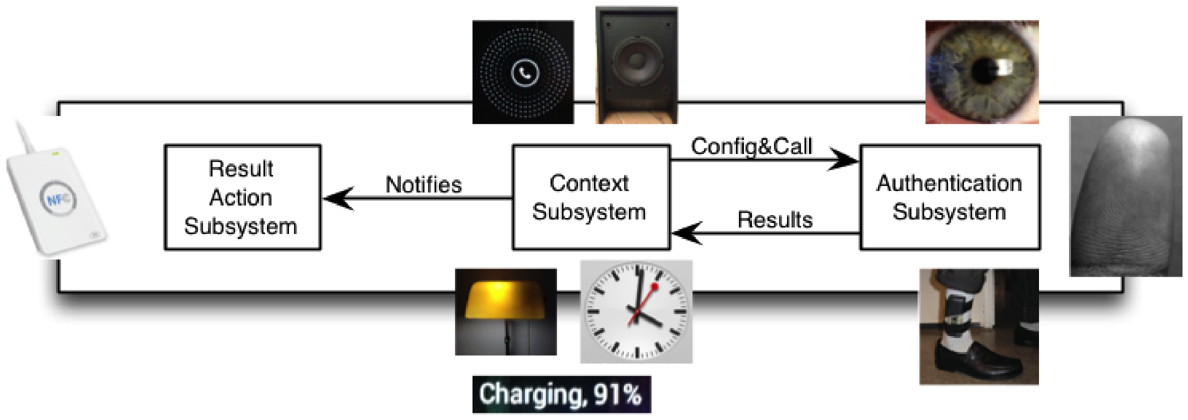


Figure 4.1.: Context-Aware Mobile Biometric System Overview

The components of the context-aware mobile biometric system are described in detail in the following subsections.

4.3.1. Context Subsystem

A context model is a set of contextual information, that describes the situation of an entity and can contain behavioural characteristics. Smartphones embed a multitude of sensors that can be used to collect a broad range of information about the environment and the behaviour of a subject. The context subsystem, which is illustrated in Figure 4.2, uses sensors embedded in smartphones to collect contextual data and, thus, represents the core of the context-aware system.

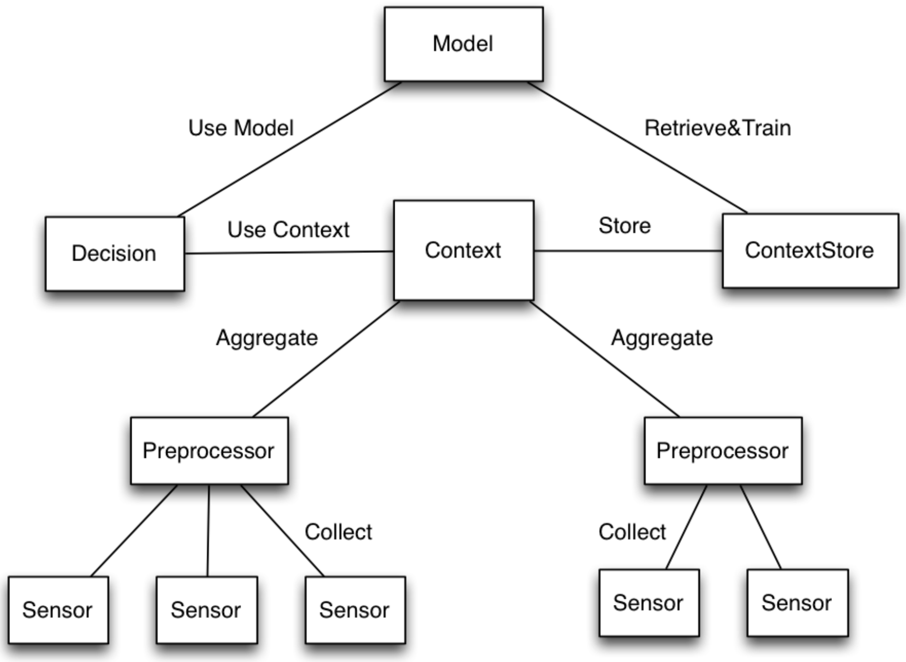


Figure 4.2.: Context Subsystem Overview

The sensor concept as outlined in the Android developer documentation is extended for the sensor components of the context subsystem, in order to enable the definition of collection periods and durations. The integration of the sensor and preprocessing components is based on the observer pattern [JGVH95], thus, sensors push data to their observers in predefined intervals. Since the observer pattern is the fundamental concept for the communication between all layers of this subsystem, it is depicted in Figure 4.3. This may be extended by offering a pull API, to obtain sensor data at arbitrary times. Sensors are grouped within a preprocessing element, which receives raw data from the sensors and performs a transformation, normalisation, or feature extraction. Subsequently, the resulting features are sent to the context component which aggregates the preprocessed data and combines it into a context data frame.

Context data frames can then be written to a persistent store for later reference. These context stores may be implemented on the device, for instance by using an SQLite database[9], or remotely on a server. Finally, the model component employs the collected

[9]`http://www.sqlite.org`

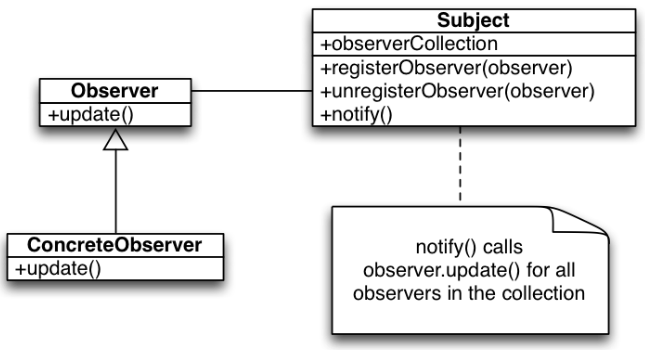

Subject
+observerCollection
+registerObserver(observer)
+unregisterObserver(observer)
+notify()
Observer
+update()
ConcreteObserver
+update()
notify() calls observer.update() for all observers in the collection

Figure 4.3.: Observer Pattern

context frames and a set of training examples, in order to generate a model of the subject. Subsequently, the current context is evaluated by the decision component, which computes the probability of the current context frame being associated with the enrolled subject.

Detailed component specifications of the proposed system are provided within the next subsections.

4.3.1.1. Sensor Component

Sensors constitute the lowest layer of the context subsystem. A sensor can measure anything, from an arbitrary physical quantity to the keyboard input of a subject. All sensors request a *WakeLock* in order to keep receiving events from the *SensorEventListener* when the device is put to sleep. The *WakeLocks* are managed by the context component, which is discussed later. The Java interface of the *ContextSensor* implementation is depicted in Figure 4.4.

Aside from methods required for implementing the observer pattern, the interface includes methods to set polling intervals and durations. When such values are set, the sensor activates itself at the given interval by registering an *Alarm* with the *AlarmManager* in the `startPolling` method. At the time the `activateSensor` method is called, a *Timer* with the specified duration is registered. A *ContextSensor* stops collecting when the *Timer* is fired. Subsequently, the collected data is passed to all registered preprocessor components. The remaining methods return sensor specific information such as a description, type, category, or information provided by the Android API such as resolution, delay, and range.

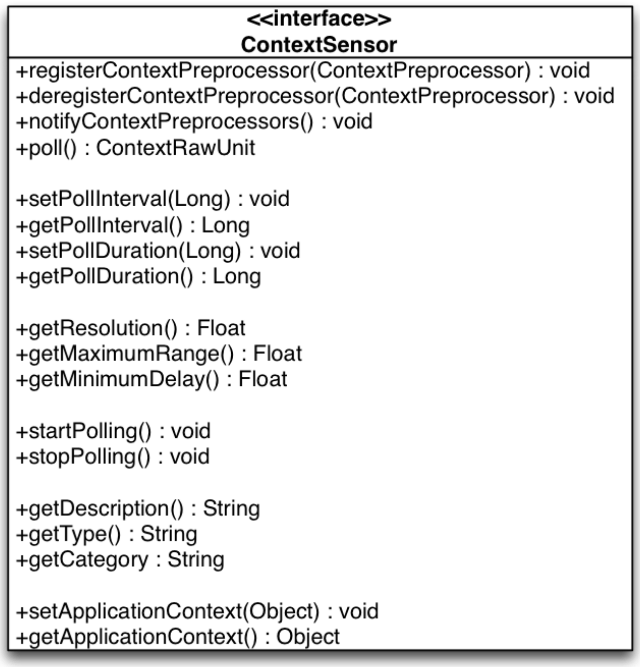

<<interface>>
ContextSensor
+registerContextPreprocessor(ContextPreprocessor) : void
+deregisterContextPreprocessor(ContextPreprocessor) : void
+notifyContextPreprocessors() : void
+poll() : ContextRawUnit
+setPollInterval(Long) : void
+getPollInterval() : Long
+setPollDuration(Long) : void
+getPollDuration() : Long
+getResolution() : Float
+getMaximumRange() : Float
+getMinimumDelay() : Float
+startPolling() : void
+stopPolling() : void
+getDescription() : String
+getType() : String
+getCategory : String
+setApplicationContext(Object) : void
+getApplicationContext() : Object

Figure 4.4.: Interface of the ContextSensor Component

4.3.1.2. Preprocessor Component

A preprocessing component can be used to create arbitrary groups of sensors. It creates instances of sensors, registers itself as an observer, and sets the polling interval and duration parameters. When a sensor finishes collecting data, it notifies the preprocessing component, which in turn calls the `poll` method of the sensor and receives a *ContextRawUnit*. This unit includes the raw sensor data and a timestamp. Subsequently, *ContextRawUnits* are preprocessed and the result is stored in a *ContextUnit* object, which in turn is passed to the context component.

Instead of implementing the preprocessing methods directly, the visitor pattern [JGVH95] is used to group the functionality within a class. Each *ContextRawUnit* has an `accept` method, which accepts a visitor object. The *ContextRawUnit* then calls a method of the visitor object, which can handle the specific data, e.g. an accelerometer raw unit would call the `visitAccelerationUnit` method of the visitor object. This enables the grouping of related functionality in the visitor object, instead of spreading the code across all *ContextRawUnit* implementations. Since different types of sensors may be used and grouped into multiple preprocessing components, the visitor pattern enables a better maintainability of preprocessing code. The interface of the preprocessing component is depicted in Figure 4.5.

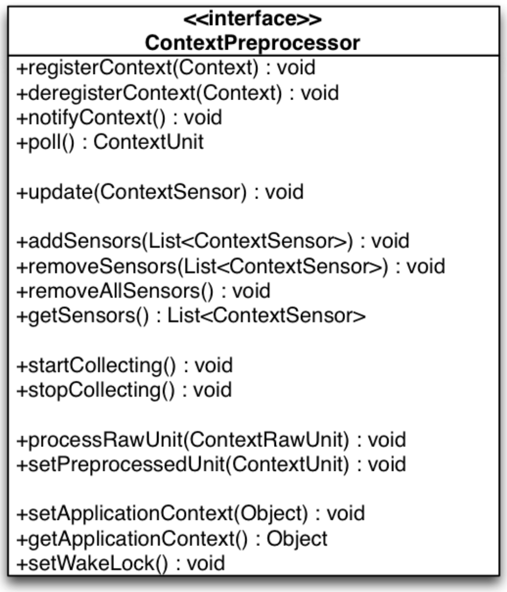

<<interface>>
ContextPreprocessor
+registerContext(Context) : void
+deregisterContext(Context) : void
+notifyContext() : void
+poll() : ContextUnit
+update(ContextSensor) : void
+addSensors(List<ContextSensor>) : void
+removeSensors(List<ContextSensor>) : void
+removeAllSensors() : void
+getSensors() : List<ContextSensor>
+startCollecting() : void
+stopCollecting() : void
+processRawUnit(ContextRawUnit) : void
+setPreprocessedUnit(ContextUnit) : void
+setApplicationContext(Object) : void
+getApplicationContext() : Object
+setWakeLock() : void

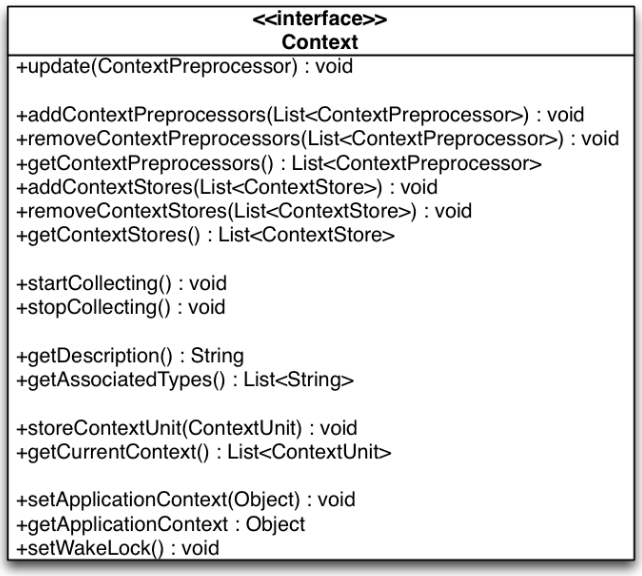
Figure 4.5.: Interface of the ContextPreprocessor component

4.3.1.3. Context Component

The context component represents the current context as a list of *ContextUnits*. Preprocessed sensor data can be aggregated by the context component and made persistent by sending the context data frame instance to a *ContextStore*, in addition to storing the last one in a buffer. The interface of the context component is depicted in Figure 4.6.

Aggregating the data to a specific format is an application-specific feature and, thus, not part of the interface design. For reasons that are outlined in chapter 6, all features of the context model need to take on numeric values for the purpose of the presented thesis. An instance of the application-specific representation of the context model can be retrieved by calling the `getCurrentContext` method.

The context component is also responsible for releasing the *WakeLocks*, which were acquired by the sensor components. References to the WakeLocks are stored in a queue. When a *ContextUnit* is received by the context component, it retrieves the head of the queue and releases the *WakeLock*. It is not important whether the released *WakeLock* was acquired by the sensor that delivered the *ContextUnit*, since only the release of the last acquired *WakeLock* will permit the device to go to sleep. This ensures that all *ContextUnits* can be received and persisted by the context component.



```
<<interface>>
Context
+update(ContextPreprocessor) : void

+addContextPreprocessors(List<ContextPreprocessor>) : void
+removeContextPreprocessors(List<ContextPreprocessor>) : void
+getContextPreprocessors() : List<ContextPreprocessor>
+addContextStores(List<ContextStore>) : void
+removeContextStores(List<ContextStore>) : void
+getContextStores() : List<ContextStore>

+startCollecting() : void
+stopCollecting() : void

+getDescription() : String
+getAssociatedTypes() : List<String>

+storeContextUnit(ContextUnit) : void
+getCurrentContext() : List<ContextUnit>

+setApplicationContext(Object) : void
+getApplicationContext : Object
+setWakeLock() : void
```

Figure 4.6.: Interface of the Context Component

4.3.1.4. ContextStore Component

A *ContextStore* provides methods to persist individual or aggregated *ContextUnits*, either directly as objects or alternatively in a JavaScript Object Notation (JSON)[10] or any other text-based representation. It can be implemented locally, e.g. by using an SQLite database, or remotely on a server. For the purpose of the data collection, which is described in chapter 5, a remote ContextStore with a write-only policy is implemented.

In addition to `insert` methods, a `select` method is provided as part of the interface, which is depicted in Figure 4.7. Since the concept of a *ContextStore* is not bound to a database implementation, it does not provide a specification for a query language. Custom domain specific languages (DSLs) for queries must be provided by the application developer. The result of a query is a list of *ContextUnits*, either individually or aggregated, depending on the application.

A *ContextStore* is used by the context and model components of the context-aware system. The interface of the *ContextStore* component is depicted in Figure 4.7.

4.3.1.5. Model Component

Context data is used to build subject-specific models in the presented thesis. Building a statistical model requires input data for the training of e.g. a regression function or a

[10] `http://www.ietf.org/rfc/rfc4627.txt?number=4627`

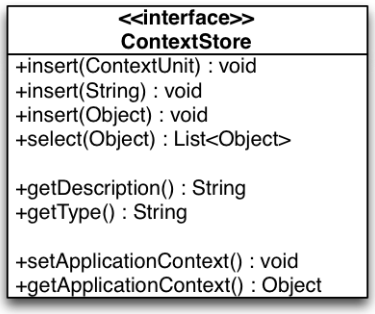

<<interface>>
ContextStore
+insert(ContextUnit) : void
+insert(String) : void
+insert(Object) : void
+select(Object) : List<Object>
+getDescription() : String
+getType() : String
+setApplicationContext() : void
+getApplicationContext() : Object

Figure 4.7.: Interface of the ContextStore Component

classifier. The *ContextStore* component provides the necessary interface to a database and is used by the model component for retrieving the training data to train a statistical model.

Since the behaviour of subjects varies over time, the system shall adapt to these changes. Training a model once and using it for all future classifications would decrease the confidence in the classification results after some time. Thus, a model needs to be trained frequently based on the most current data. An optimal model component updates current models with the latest data, which is referred to as the online setting. However,the implementation of this component in the presented thesis requires a complete training set, which is referred to as the batch or offline setting. In either case, these details are implementation-specific and, thus, not part of the interface, which is depicted in Figure 4.8.

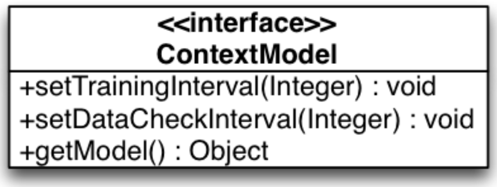

<<interface>>
ContextModel
+setTrainingInterval(Integer) : void
+setDataCheckInterval(Integer) : void
+getModel() : Object

Figure 4.8.: Interface of the ContextModel Component

The interface of the *ContextModel* component includes a method to retrieve the trained model. Since the design is not bound to a specific type of models, the return value of this method is a generic object. In addition, the training interval can be set by specifying a duration in hours, in order to enable the adaptive behaviour of the system. Another timer can be set, that specifies the interval at which the database is queried to check for new data. The interval for the data check timer shall be smaller than the interval for the training timer. If not enough new data is available when the training timer is fired, a flag shall be set. This flag is checked in the method called by the data check timer. If enough data is available and the training flag is set, the method called by the data check timer shall start the training process and reset the training timer. This process ensures, that the training is still executed when the collection or storage of new data is slightly delayed. However, when the data is still missing when the next training timer is fired, the flag is cleared and the process starts over. An activity diagram for the training timer workflow is depicted in Figure 4.9 and the activity diagram for the data check timer workflow in Figure 4.10.

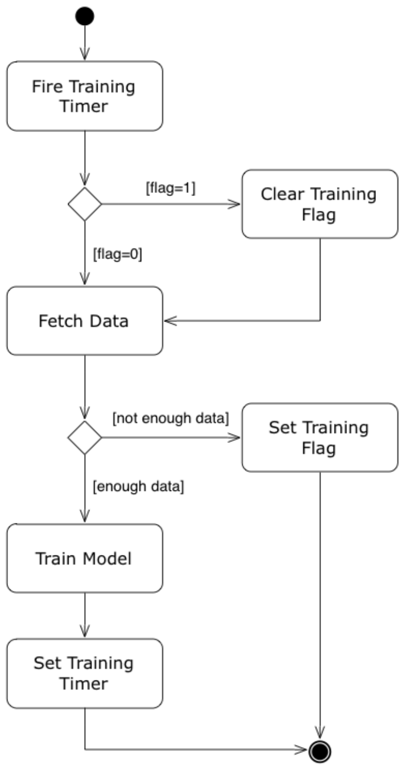


Figure 4.9.: Activity diagram for the training timer workflow

4.3.1.6. Decision Component

When a statistical model has been built, the decision component retrieves the current context from the context component and the trained model from the model component in configurable intervals, or on request. It then uses the model to classify the current context and derives a confidence score, that reflects the strength of the association between the current context and the enrolled subject. The confidence score can be mapped to an

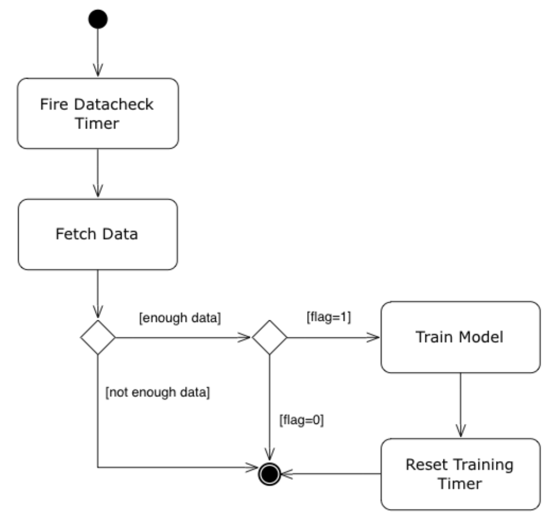


Figure 4.10.: Activity diagram for the data check timer workflow

application-specific security level, that determines how many authentication factors shall be required. An example for such a mapping is given in Table 4.2.

Table 4.2.: Example for a context security level to authentication factors mapping

Context Security Level	**Confidence**	**Authentication Factors**
3	>=99%	0
2	80%–98%	1
1	60%–79%	2
0	<60%	3

The context security level increases with the confidence score. If the confidence of the context classification is low, additional authentication factors are required to keep the overall security level of the system high. Based on this mapping, the decision component controls the authentication flow by calling the respective methods in the authentication subsystem. The interval at which the current context is classified can be configured through the interface of the decision component. The decision interval shall be greater or equal to the interval at which the data for the training set was collected.

In addition to controlling the authentication subsystem, the decision component also receives the authentication results and propagates these to all registered observers. Observers of the decision component comprise the result action subsystem, which is described later. The interface for the decision component is depicted in Figure 4.11.

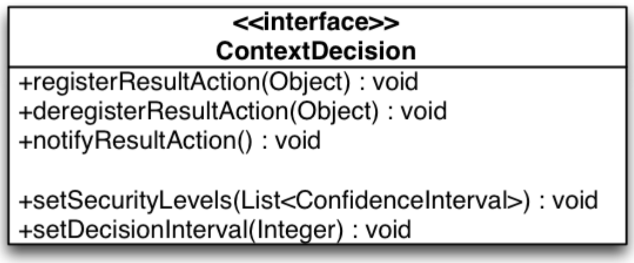

<<interface>>
ContextDecision
+registerResultAction(Object) : void
+deregisterResultAction(Object) : void
+notifyResultAction() : void
+setSecurityLevels(List<ConfidenceInterval>) : void
+setDecisionInterval(Integer) : void

Figure 4.11.: Interface of the Decision Component

4.3.2. BioAPI-based Authentication Subsystem

The BioAPI has been established as a standard for application programming interfaces in biometric systems. Originally designed for the C programming language, an extension to the standard defines interfaces for object oriented languages [ISO12b]. This international standardisation project addresses a specification for the Java programming language, which is the default language for the Android operating system, and thus a natural fit for the presented context-aware system. A BioAPI compliant system is comprised of the following components:

- BioAPI Framework
- BioAPI Units
- Biometric Service Provider (BSP)
- Biometric Function Provider (BFP) – optional

BioAPI Units are categorised as: sensor units, archive units, matching units, processing units, and can be implemented by BSPs or externally provided in BFPs.

BSPs provide an interface for the enrolment, authentication, and identification of a subject. Information on the registered components can be retrieved from the BioAPI framework component. Given the information on installed BSPs, an extension of the context subsystem can be utilised to automatically determine an authentication module which is expected to perform best under the current conditions. Figure 4.12 depicts the entire authentication subsystem.

While the BioAPI defines interfaces for the components of a biometric system and a component registry for the management of these, the authentication flow is determined by the application. In contrast to MBASSy (see section 4.2), there is no way to specify an

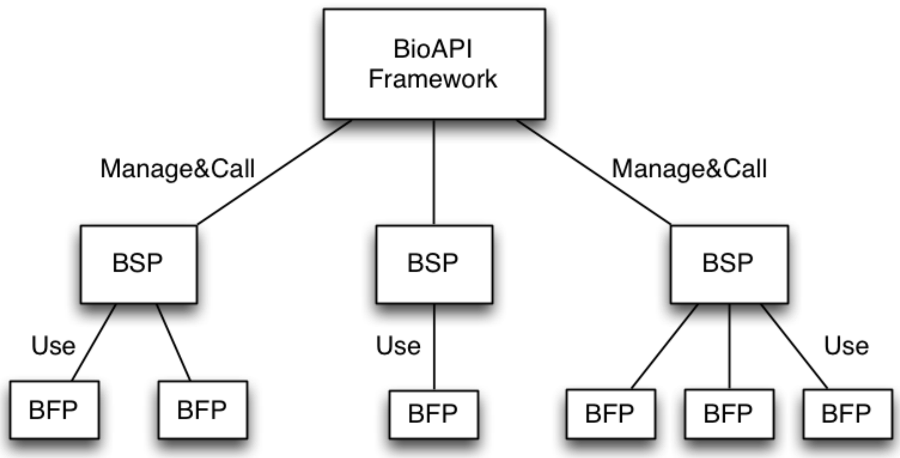


Figure 4.12.: Authentication Subsystem Overview

execution order for the registered components. This functionality must be implemented in the decision component, by calling the `enumBSPs` method of the BioAPI framework and commit to a selection of desired authentication modules in the source code of the application. However, it is possible to expose this functionality through a user interface to let a user make these choices manually. Only authenticated subjects may alter the configuration after the initial setup. Since the context-aware system does not provide such a user interface, but rather a framework that can be utilised by developers, additional steps are required to develop an application like MBASSy for demonstration purposes. Thus, MBASSy is an application that can be based on the context-aware system presented in this thesis.

4.3.3. Result Action Subsystem

At the time an authentication result is available, the system shall provide a way to handle it. The decision component of the context subsystem implements an interface that other software components can attach to, in order to register for authentication result notifications. These components can, for instance, propagate the results to objects in the proximity of the user (see Figure 4.1). Notifications are published by the decision component to its subscribers. Again, the underlying design is based on the observer pattern (see Figure 4.3), i.e. the result action subsystem is comprised of software components that handle authentication results. In addition to the authentication decision, a confidence score may be returned that can be utilised by components like policy-enforcers, which may require a mapping of scores to security levels.

An example for a result action was implemented by Derawi et al. [DWMB12]. They combined the recognition of finger images, taken with a smartphone camera, with token-based authentication over Near Field Communication (NFC). In case the finger image of a subject was successfully verified, the smartphone sent an authentication token over NFC to a NFC reader at a door, which was connected to a PC inside the room. The received token was compared against a list of tokens on the PC. In the event of a match between the authentication token and an item of the access control list, an electronic door lock was activated to grant access to the room. A generalised sequence diagram of this use case is depicted in Figure 4.13.

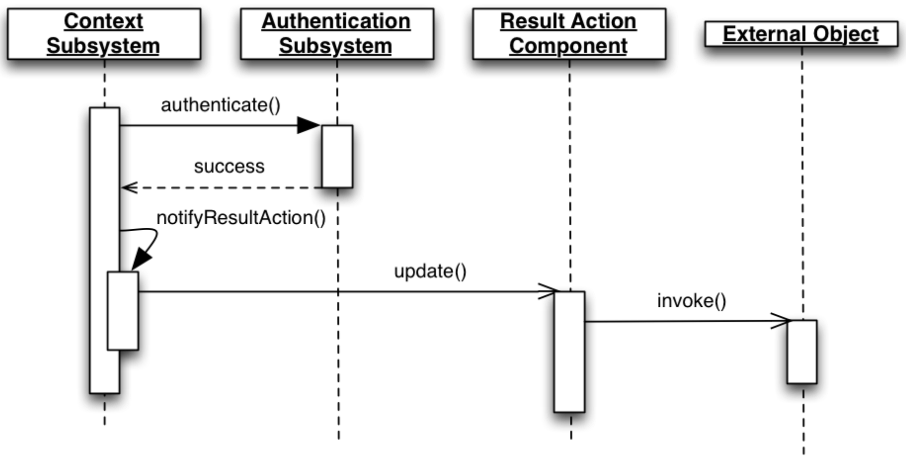


Figure 4.13.: Sequence diagram of the result action workflow

The result action subsystem is designed to simplify the development and deployment process for such use cases by providing an interface, which allows additional components to subscribe to authentication decisions. Developers can focus on implementing the action, without having to design and implement proprietary interfaces to the context subsystem.

4.4. Implementation Aspects

Not all of the discussed components are implemented as part of the presented thesis. A simplified version of the *ContextStore* is developed for the data collection, which is described in chapter 5. The *ContextStore* uploads data to a server, but has no methods to query the server-side database, since this would require server-side software that is not required for the collection purpose. Communication with a server is optional for the general use case, but was necessary for the data collection purpose of the presented thesis.

The model and decision components are not implemented as described, because the evaluation and testing is performed on a desktop computer. Data preparation, training and evaluation code is written in the R programming language[11], which makes prototyping easier and more efficient.

Since the authentication subsystem is fully described by the BioAPI standard, an implementation of it would not add much value to the objectives of the presented thesis. A reference implementation of the Java BioAPI[12] already exists, so that an implementation of the authentication subsystem mainly requires the development of Android specific functionality.

However, all parts that are required for the data collection are implemented. This includes the context component, preprocessors, sensors and the simplified database.

A topic that is not addressed by the interface specifications is the inherent concurrency when using multiple sensors. The preprocessing components buffer the raw data that they poll from the sensors before notifying the context component. When the context component polls data from the preprocessor, a conflict can occur when trying to modify a buffer that is not synchronised. If an element is removed from such a buffer while an iterator uses it, a *ConcurrentModificationException*[13] is thrown. In order to avoid synchronisation problems, preprocessor components shall use thread-safe containers, for instance a *LinkedBlockingQueue*[14]. If the context component aggregates data units of all registered sensors before storing a combined feature vector in the *ContextStore*, then it shall also implement a thread-safe container for the buffering.

[11]`http://www.r-project.org`

[12]`http://sourceforge.net/projects/bioapijava/`

[13]`http://docs.oracle.com/javase/7/docs/api/java/util/ConcurrentModificationException.html`

[14]`http://docs.oracle.com/javase/7/docs/api/java/util/concurrent/LinkedBlockingQueue.html`

5. Data Collection

An important part in the process of developing a context-aware system is the feature extraction from available signals. A data collection was carried out as part of this work, in order to statistically analyse various sensor signals and determine which features carry the most valuable information about a subject in terms of context-modelling.

This chapter contains information on the collection process, the collected raw features, privacy aspects, and challenges along the way. Obtained results are presented in the last subsection.

5.1. Collection Process

The collection of context data in order to build a model of a subjects' daily activities requires time. Ideally, two weeks of contextual information per subject shall be collected for the purpose of this thesis. Using a small set of devices and handing them to one subject after another would either put a limitation on the maximum number of subjects or the duration of each individual collection phase. Thus, the data collection has to be performed on the personal devices of the subjects to ensure scalability.

To achieve this goal, a data collection application for the Android operating system, which is referred to as ContextCollector, is developed. Before a subject can start the collection, a questionnaire has to be filled in. The following information is requested:

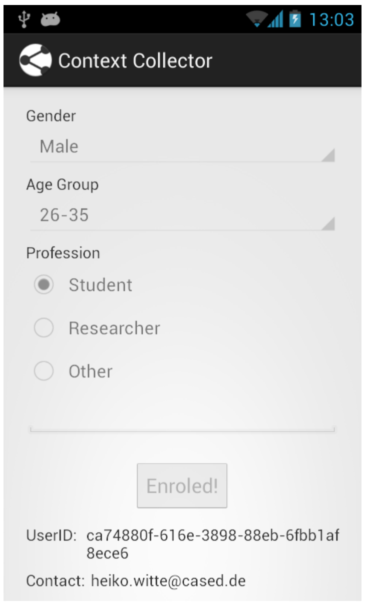
- Gender (male, female)
- Age group (<18, 19–25, 26–35, 36–45, 46–55, >55)
- Profession (Student, Researcher, Other)

Subjects enrol by filling in the questionnaire and accepting the terms and conditions. The following additional information is sent to the server:

- Device model

- Android version
- User ID

The user interface of the application is depicted in Figure 5.1.



Gender
Male

Age Group
26-35

Profession
[✓ Student]
Researcher
Other

UserID: ca74880f-616e-3898-88eb-6fbb1af8ece6

Contact: heiko.witte@cased.de

Figure 5.1.: User interface of the ContextCollector application

After enrolment, the application starts to acquire sensor data in twenty minute intervals. Each sensor collects data for ten seconds. The collected data is then preprocessed, transformed to a JSON representation, an sent to a server over a Secure Sockets Layer (SSL) connection. The server application is based on Ruby[1] and MongoDB[2], which is a schema-free and document-based NoSQL database[3]. MongoDB is chosen as it can handle heterogenous data and directly insert JSON formatted strings of text.

The server verifies the validity of a request by a successful parsing of the input into a dictionary. Enrolment requests are stored in the "user" collection, while requests containing contextual units are stored in the "data" collection of the database. If the structure of the received JSON is invalid, i.e. it cannot be parsed into the dictionary, the received packet is discarded.

A simplified version of the collection process, that omits the collection intervals and durations, is depicted in Figure 5.2.

[1]`http://www.ruby-lang.org/en/`
[2]`http://www.mongodb.org`
[3]`http://www.christof-strauch.de/nosqldbs.pdf`

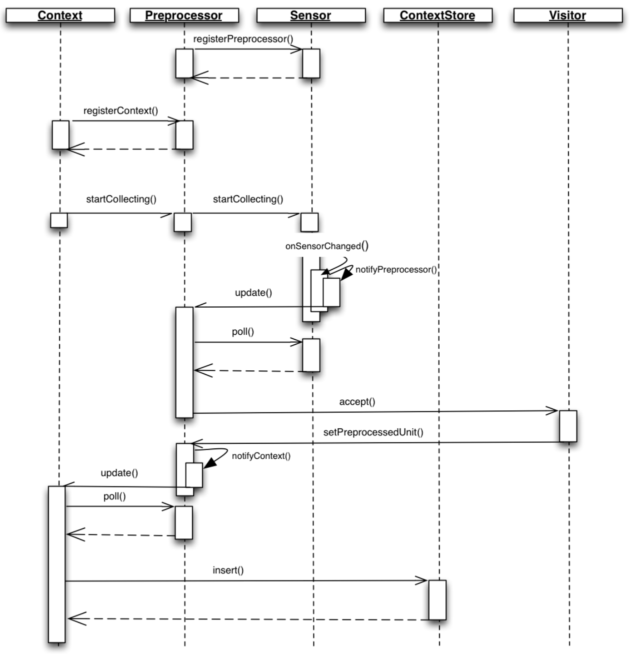


Figure 5.2.: Sequence diagram of the collection workflow without collection intervals and durations

5.2. Collected Data

Smartphone devices include a multitude of sensors that measure a variety of environmental aspects, i.e. contextual information can be easily acquired. In addition to generic sensors that measure physical magnitudes, quantities such as the current time, network traffic, battery percentage, etc. can be analysed, representing a second type of sensor measures. The former type of sensors is referred to as physical sensor, the latter as logical sensor. Furthermore, a discrimination between active and passive sensors is necessary. Active sensors can be activated and deliver data in predefined intervals, while passive sensors only deliver data in case a subject interacts with the device. An example for a passive-logical sensor is the measurement of activations and deactivations of a devices' screen.

Sensors send the acquired data (together with a timestamp) to the preprocessing element (see Figure 4.2) which can perform a normalisation, transformation, reduction, filtering, or mapping of the data in order to extract features. Sophisticated algorithms may be leveraged to extract higher-order features, instead of trivial statistical ones. An example is the use of raw accelerometer data as input for an activity recognition algorithm [WN11]. The derived activity may then be represented as a discrete value.

Preprocessing of sensor data in the presented configuration of the system is limited to a reduction of the data by computing the mean and median of a set of raw sensor values. This simplification is imposed in order to keep the focus on system design, development, and evaluation. The employed types of sensors are summarised in Table 5.1.

Table 5.1.: Different types of employed context sensors (P=physical, L=logical, A=active, PS=passive)

Sensor	**Type**
Location	P,A
Accelerometer	P,A
Magnetic Field	P,A
Microphone	P,A
Light	P,A
Battery	L,A
ScreenState	L,PS
ShutdownBoot	L,PS
Call	L,PS

A detailed discussion of the included types of sensors is given in the next subsection.

5.3. Sensor Overview

This subsection contains a detailed discussion of the sensors included in the data collection (see Table 5.1). In addition to a description of the collected features, sensor-specific information with regard to functionality and calibration aspects is provided where appropriate.

5.3.1. Location

Location sensors are used to determine position coordinates and supplemental information. Mobile devices have access to two different kinds of location providers:

Network provider This location provider uses cell identifiers and WiFi SSIDs to query a database which delivers the locations of the mobile cell or WiFi router. The position estimate is computed based on signal strength of the surrounding cells and routers by triangulating the signals. The accuracy of network location providers depends on the distance between cell towers or routers. Thus, better performance is achieved in cities with a tight grid of cell towers and WiFi networks.

Global Positioning System (GPS) provider This provider delivers the most accurate position estimate with an accuracy of up to ten meters. It is not guaranteed that the mobile device can find a satellite signal and the search for it can take up to a few minutes. Location fixes from the network provider can be used to speed up the search. Thus, it takes more time to receive position estimates from this location provider. A GPS sensor also requires a lot of power and thus reduces battery life significantly.

Depending on the type of location sensor that is used, different kinds of data are delivered. While the network providers are limited to a position estimate which is returned as a pair of latitude and longitude coordinates, GPS provides additional information like altitude and speed. Table 5.2 provides an overview of which data is collected by the ContextCollector application.

Table 5.2.: Location Data

Data Field	Data Type	Description	Location Provider
Latitude	float	Coordinate	Network and GPS
Longitude	float	Coordinate	Network and GPS
Accuracy	int	m	Network and GPS
Altitude	float	m	GPS
Speed	float	$\frac{m}{s}$	GPS

The mobile application receives updates of both location providers and keeps the location fix with the best accuracy. The rational behind collecting location information is to determine the places a subject visits at a given day and time.

5.3.2. Accelerometer

Accelerometers in smartphones measure accelerations of the device on three axes. These sensors were initially used to automatically adjust the user interface when the device is rotated [App13]. They can also be used as an input device to control games or other applications by moving the device.

In previous work of the da/sec Biometrics and Internet Research Group[4] at CASED, research was carried out on using accelerometers for biometric gait recognition [NBRM11; NBB11; NWB12]. The acceleration signal is recorded, normalised, segmented and subsequently used to train Hidden Markov Models (HMM) [RJ86] and Support Vector Machines (SVM) [Vap82]. The resulting classifiers were used in authentication processes to authenticate subjects. For this purpose, a large amount of raw accelerometer data was acquired for the training process.

Raw data is not collected in the presented thesis in order to save memory and bandwidth, and to focus on developing multiple sensors and the overall architecture of the system. Instead, the mean of the raw data is computed for each axis for a set $X = \{x_1, x_2, \ldots, x_n\}$ of n collected values (where $n \approx SensorFrequency * 10$):

$$mean(X) = \frac{\sum\limits_{i=1}^{n} x_i}{n} \tag{5.1}$$

Since the mean is sensitive to outliers, the median is computed as well:

$$median(X) = \begin{cases} x_{\frac{n+1}{2}}, & \text{if } n \text{ is odd} \\ x_{\frac{n}{2}}, & \text{else} \end{cases} \tag{5.2}$$

The rational for this reduction is given in section 5.2.

By reducing the data set and only using the locational information of summary statistics, sophisticated algorithms for activity recognition cannot be performed. Instead, a heuristic is applied, by assuming physical activity whenever the sum of the accelerations of all axes exceeds the gravitational force of $9.81m/s^2$ by a certain threshold. The collected fields are listed in Table 5.3.

Table 5.3.: Acceleration Data

Data Field	Data Type	Description
median X,Y,Z	float	$\frac{m}{s^2}$
mean X,Y,Z	float	$\frac{m}{s^2}$

[4]http://www.dasec.h-da.de

5.3.3. Magnetic Field

A magnetic field sensor measures the magnetic field of the earth on three axes in micro Tesla (μT), which is a unit for magnetic flux density. A majority of the available smartphones include such a sensor. It is primarily used to provide additional information in mapping applications, so that an arrow can be drawn on a map of the current location, that reflects the direction in which the device is headed.

The magnetic field sensor can be used in combination with the accelerometer to determine the current position relative to the magnetic North Pole. Sensor readings are influenced by metallic objects, electronics and the earth magnetic field itself and thus, are context-dependent.

A drawback of this sensor is, that it needs to be calibrated from time to time. This is normally done by waving the phone around in an eight-figure, which triggers recalibration. Since the required user interaction contradicts the goals of the presented thesis, the sensor remains possibly uncalibrated throughout the entire data collection. Thus, the aforementioned computation of a position relative to the magnetic North Pole is not performed, as the correctness of the underlying data cannot be guaranteed.

The collected data is reduced by computing the mean and median. An overview is given in Table 5.4.

Table 5.4.: Magnetic Field Data

Data Field	**Data Type**	**Description**
median X,Y,Z	float	μT
mean X,Y,Z	float	μT

5.3.4. Microphone

Data recorded by microphones can be analysed and are used to authenticate subjects based on their voice characteristics.

For the context model in the presented thesis, the noise levels shall be collected. A method to retrieve the maximum amplitude of the recorded samples is offered by the Android API for the microphone. Instead of raw samples, the mean and median of these maximum amplitude values are computed and stored in the database. No physical unit is associated with the amplitude values, which have a range of 0 to 32767. A microphone converts pressure (Pascal) to Volts and the maximum amplitude values can be understood as a 16 bit discretisation of the volume.

Unfortunately, the discretisation results vary among different vendors or microphone models. Again, a calibration is necessary to obtain reliable results. As the goal of this sensor in the context model is to give a clue about the noise level, the calibration is not necessary. A comparison of data recorded by different devices without the calibration step has to take this into account.

An overview of the collected data is given in Table 5.5.

Table 5.5.: Microphone Data

Data Field	Data Type	Description
median amplitude	float	value between 0 to 32767
mean amplitude	float	value between 0 to 32767

5.3.5. Light Sensor

Ambient light sensors are used in smartphones to automatically adjust the screen brightness to ambient light levels, or to turn off the screen in case a user is holding the device to his/her ear during a phone call.

The unit of the data delivered by the light sensor is lux (lx), which measures luminous flux per unit area. Since the sensor is factory-calibrated, no further calibration steps are necessary and the raw data can be used directly. Illuminance data carries information about the environment a subject uses the device in, as well as behavioural characteristics, e.g. when a subject is keeping the device in a pocket or bag at the time of measurement.

Again, the mean and median of the illuminance values are stored in the database. An overview of the collected data is given in Table 5.6.

Table 5.6.: Ambient Light Data

Data Field	Data Type	Description
median brightness	int/float (depends on device type)	lx
mean brightness	int/float (depends on device type)	lx

5.3.6. Battery

The battery sensor is a logical sensor. Collected data includes the battery percentage and a charging indicator at measurement time.

Monitoring battery and charging status gives an insight into how intensely a subject uses a device and the charging behaviour. Some people may carry their chargers with them all the time or charge their devices every time they encounter a free USB port on a computer. Others may not mind draining their batteries and operate their device on lower average battery percentages.

A direct comparison of the data collected on different devices has to take the quality of the battery and the operating system version into account. The energy consumption of a device is heavily influenced by factors, which do not correlate to user behaviour. Thus, data about the device model and operating system version is collected as part of the enrolment process. Additional information on the energy consuming processes would deliver even more insight. An indicator for user driven energy consumption is, when the display is among the top energy consumers. However, this information is not part of the data collection in the presented thesis.

An overview of the collected data is given in Table 5.7.

Table 5.7.: Battery Data

Data Field	**Data Type**	**Description**
battery percentage	float	current charge of the battery in percent
charging status	boolean	indicates if the device is currently charging

5.3.7. ScreenState

Another logical sensor is the screen state sensor. Every time a subject activates the display of a device, the timespan until display deactivation is measured. Deactivation may be triggered by the user pressing a button, or by a timeout defined in the devices system settings.

Some people may constantly check news, websites, social media, or other online resources on their smartphones, while others might only use the phone and texting features of their device. This data might also serve as a corrective for the battery sensor data, as it measures user activity. Besides the sheer volume of screen activations, the part of the day where the activity occurs may be relevant. Since all sensor units include a timestamp so that this property can be exploited.

An overview of the collected data is given in Table 5.8.

Table 5.8.: Screen State Data

Data Field	**Data Type**	**Description**
activation	String	timestamp of the display activation
deactivation	String	timestamp of the display deactivation

5.3.8. ShutdownBoot

This logical sensor saves a timestamp for the shutdown and boot of a device. Information is stored in the database in the event of a boot.

Given the shutdown and startup times of a device, usage patterns can be derived. For instance, while some users may shutdown their device before going to sleep and turn it on in the morning, others may leave the device running indefinitely. Some may also restart their device regularly in order to clean up memory. Lots of scenarios are possible, so this data shall give insights into this specific user behaviour. Additionally, by combining this information with the battery sensor, it can be inferred if the device went off because of a low battery charge.

An overview of the collected data is given in Table 5.9.

Table 5.9.: Shutdown and Boot Data

Data Field	**Data Type**	**Description**
boot	String	timestamp of the boot
shutdown	String	timestamp of the shutdown

5.3.9. Calls

Learning when a subject makes or receives calls and the duration of those calls, gives insight into the social or professional behaviour. Among the collected data is the information if the current call target or source number is stored in the subjects' contacts list.

While it cannot be inferred, whether the call is made in a personal or professional context, based on this information, it can be assumed that the respective person or institution is of some significance to the subject.

An overview of the collected data is given in Table 5.10.

Table 5.10.: Calls Data

Data Field	**Data Type**	**Description**
duration	int	in seconds
direction	int	0 incoming, 1 outgoing
number known	boolean	indicates if the calling or called number is in the contacts list

5.3.10. WiFi

A hash of the WiFi SSID is stored if a subject is logged onto a network at the time of measurement. This information seems unnecessary or redundant at first, given that a network is always associated with a location and the location providers even use this data to determine a location fix. At second glance, there are cases where this information can be of help in determining the security level. When a device is lost or stolen, picked up by someone and subsequently logged onto an unknown network, this can be classified as an anomaly. The unknown network does not need to be in proximity of the usually used network, since the location information is evaluated separately. Even if the subject willingly logs on to foreign networks, the event should be classified as anomalous.

As WiFi developed into a mainstream technology years ago, a multitude of networks may exist at the same physical location. Thus, the physical position which often is inaccurate can be augmented by using network information.

An overview of the collected data is given in Table 5.11.

Table 5.11.: WiFi Data

Data Field	**Data Type**	**Description**
SSID	String	SHA–256 hash of the SSID

5.4. Behavioural Characteristics

According to the harmonised biometric vocabulary [ISO12a], a behavioural characteristic is understood as physiological or behavioural processes created by the body. A common behavioural characteristic is keystroke-dynamics, which measures the typing behaviour of a subject. This characteristic fits with the definition, since it directly relates to the human hand. It can be safely assumed though, that human behaviour is still a unique property of a subject, even in case it does not directly relate to a visible part of the body. Thus, the definition seems a bit narrow. Behavioural features collected in the presented thesis mainly relate to the subject as a whole and not necessarily to a single body part. To fit with the definition though, behaviour can be regarded as the result of a thought or learning process and thus, directly relates to neural structures in the human brain.

A majority of the collected features can be regarded as behavioural characteristics of a subject. For instance, data from the accelerometer indicates physical activity, the geographical location gives insight into the traveling behaviour of a subject, and the screen state sensor measures interaction with a device. Furthermore, the call sensor gives insight into the communication behaviour of subject, i.e. when calls are made and how long these usually are. Additionally, the shutdown and boot sensor indicates when a subject turns the device off and on, but it also measures automatic shutdowns in case of a drained battery. The automatic shutdown case could be discarded by simultaneously checking the current battery charge and ignore the event in case the battery is low. The microphone sensor measures the acoustic noise level, but could be modified to record sound to audio files, which could be processed in order to identify the voice of a subject and thus, the talking behaviour (frequency, duration, loudness etc). The magnetic field sensor could be used to determine the direction a subject is facing.

Some of these ideas can not be implemented as part of the presented thesis due to time constraints and thus, are subject to future work.

5.5. Privacy Aspects

When collecting personal data, an important aspect is privacy. A detailed description of the data collected and their use is specified in the terms and conditions participants had to accept as part of the enrolment. Besides stating what is collected and what it is used for, the collected data itself should only carry as much information as required for the analysis.

The protection of an individuals privacy starts with the registration process. No personally identifiable information is requested from the participants. A basic requirement is the assignment of unique identifiers to the subjects. An easy and straightforward way to do so is to use the Unique Device Identifier (UDID) of the smartphone. This solution is problematic, since everyone who has access to the device and the database can directly find the data records of a subject. To avoid such a scenario, the UDID is combined with a random number and then passed to an Android system function, that computes a Universally Unique Identifier (UUID) of the string. Additionally, demographic information of the subjects is collected (see section 5.1), as well as the device model and Android version. This data may also threaten the privacy in case that only a single subject is enrolled with a specific device model and Android version.

A harsh critique during the collection phase was, that the application uses the microphone, the fear being, that calls are intercepted and contents of the talk sent to the server. As already stated in subsection 5.3.4, only the volume or acoustic noise level is measured, even though raw audio data contains valuable and exploitable information. Furthermore, the microphone is polled in fixed intervals and not activated when a call is active (at least not intentionally).

Some participants were worried by the collection of the location feature. While several methods exist to perturb location information [SS98] by adding noise and grouping several subjects into a large enough location cluster, these methods would defeat the purpose of the data collection. Location information has to be as accurate as possible for the context-model of a subject.

Other features, like the recording of screen activations, were not criticised, presumably because these measure when an activity occurs and not what the activity involves. While it would be great to learn about the specifics of a subjects activity, it would arguably be harder to find participants willing to deliver this data.

The WiFi sensor stores a hash of the SSID instead of cleartext, which might include valuable information to identify a subject. While an attacker might use dictionary or similar attacks, this protection method is adequate for the purpose.

Another aspect is the transfer of the data from the mobile devices to the server. Data uploads only occur when a WiFi connection is available. Since a subject may well be using an unencrypted public WiFi network, the data transfers between the mobile device and the server shall be secure. To ensure privacy, the devices establish encrypted SSL connections only.

5.6. Challenges

Among the challenges, getting enough subjects to participate in the data collection was the hardest. As described in the previous section, people are worried about the amount and type of data that is collected.

Several calls for participation were sent via email, social networks (Twitter[5], Facebook[6], Google+[7], LinkedIn[8], Xing[9]) and technology forums. Additionally, posters with the call for participation were hung in the kitchens and student pools of CASED and the blackboard of the computer science department of the university.

Another non-negligible challenge of this data collection is the support of heterogenous devices. Given the requirement, that subjects have to use their personal smartphones, a wide variety of device models, vendors and Android versions had to be supported. It is a general problem in the development process for Android, that can only be solved by testing on a wide range of Android versions and device models. Only a single device (Google Nexus S) was available for the development and testing of the mobile application. This turned out to be problematic, as some software failures occurred only on a subset of the devices. Thus, software bugs had to be fixed as part of an ongoing process, instead of within a testing period.

Since the context data collection application constantly activates sensors and establishes network connections, maximising battery life is a major objective. The first versions of the application used a WakeLock to keep the CPU active, since otherwise timers and sensor callbacks are not triggered. This led to unacceptable power consumption and drained batteries fast. A solution to this problem was using the Android *AlarmManager*, that is capable of waking devices from sleep at a given time. When the device is woken up by the alarm, a *WakeLock* is set for the duration of the collection of each sensor and released afterwards. Each sensor sets and releases its own *WakeLock*.

Current data plans for smartphones have a hard limit of approximately 300MB of monthly data usage. Hence, data is only sent if a WiFi connection is available. In case the device is not logged onto such a network, the data is written to a cache. A timer is triggered every five minutes to check for the network state, sends the cached data in case the device is connected to WiFi, and deletes the data from the internal cache. An error

[5]`http://www.twitter.com`
[6]`http://www.facebook.com`
[7]`http://plus.google.com`
[8]`http://www.linkedin.com`
[9]`http://www.xing.de`

in the implementation of this functionality lead to the creation of duplicate entries that accumulated quickly and increased battery and bandwidth usage. A problem that is not addressed by the application is the switch from a WiFi to a cellular network during a transfer.

Failed data transfers shall not lead to data loss. When a transfer fails, the data is written back to the internal cache, so that it can be send in a later attempt.

5.7. Results

In the following subsections, summary statistics of the participants, used devices, Android versions and collected data are presented. Demographic and quantitative information is presented, and the quality of the data is measured in terms of completeness. An analysis of the individual features is provided in chapter 7.

5.7.1. Participants

During the enrolment, participants had to specify their gender, age group and profession. While a more diverse group of participants would be desirable, the results indicate a large homogenous subgroup. The total number of participants is 26.

A majority of the participants is male and in the age group 26–35. This distribution across age group and gender was expected, as most of the participants are computer science students at the university.

Participants could choose between “Student”, “Researcher”, and “Other” for the profession. If they selected “Other”, they could provide the job title in a text field. Most of the participants are students, and the other major group provided other profession titles.

5.7.2. Devices

As indicated in section 5.6, a multitude of devices had to be supported. Among the 26 participants were 16 unique device models (see Figure 5.3) and eight different versions of Android (see Figure 5.4).

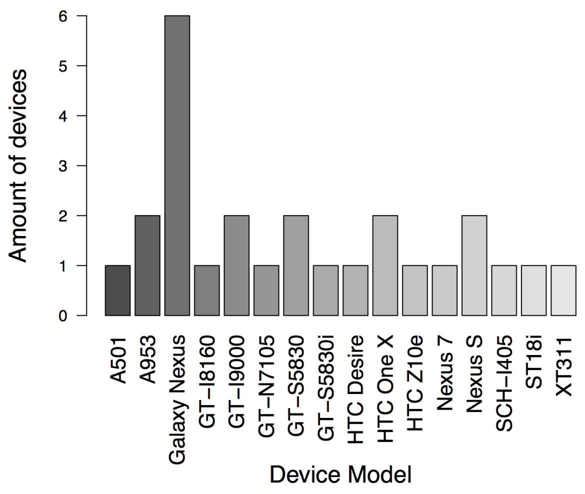


Figure 5.3.: Devices used in the data collection

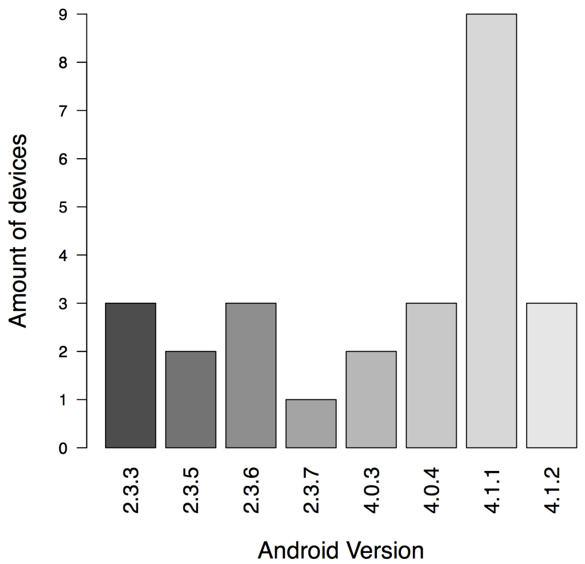


Figure 5.4.: Android versions used in the data collection

5.7.3. Data Quality

Since several errors occurred during the data collection, it is necessary to analyse the amount of lost information for each feature that is collected in fixed intervals. Features that are based on user activity cannot be evaluated for this purpose, as they are collected at arbitrary times.

Data units are grouped into the cells of a matrix, which has weekdays as columns and day sections as rows. Each data unit is annotated with a timestamp, which includes the weekday and the time. The matrix column of a data unit is determined directly by extracting the weekday from the timestamp. The section of day is computed by mapping the hour to one of the sections as described in Table 5.12.

Table 5.12.: Mapping of hour to day section

Day Section	**Hours**
1	[0, 3)
2	[3, 6)
3	[6, 9)
4	[9, 12)
5	[12, 15)
6	[15, 18)
7	[18, 21)
8	[21, 0)

The cell that includes the most data units determines the number of days the data collection was active on the device, and is used as the basis for computing the average degree of completeness in the other cells.

The average degree of completeness for each subject and feature is presented in Table 5.13. Not all subjects delivered enough data to be considered in the evaluation, thus the total number of subjects is reduced to 17. This metric is employed in order to determine how much data of a feature was lost during the collection phase. This information might be valuable when evaluating the system performance, since it provides a means to compare evaluations for different subjects based on the completeness of each feature. For instance, if a model performs poorly and the underlying data has a low average degree of completeness for the location feature, and another model performs good with a high average degree of completeness for this feature, it might be worthwhile to investigate whether the low performance of the former model is due to the lack of available location information, or to a low quality of the location data that is available.

Table 5.13.: Average relative degree of completeness per subject and feature in percent (A=Acceleration, M=Magnetic Field, Loc=Location, N=Noise, Lig=Ambient Light, B=Battery, W=WiFi)

Subject	**A**	**M**	**Loc**	**N**	**Lig**	**B**	**W**
1	66.84	71.89	50.91	23.86	64.47	63.20	63.27
2	66.87	66.63	46.01	26.67	63.37	26.40	26.90
3	67.07	66.36	61.72	3.64	0.00	66.16	65.35
4	43.64	43.77	42.96	25.91	43.85	43.97	43.89
5	82.99	82.92	78.22	16.46	82.53	73.44	73.01
6	60.10	60.27	63.17	16.36	59.64	58.79	58.41
7	69.50	71.21	70.13	4.85	0.00	67.66	67.72
8	78.34	77.43	58.10	12.73	77.87	80.24	76.13
9	63.97	63.80	55.14	6.97	64.10	63.43	63.47
10	57.06	54.00	47.27	6.97	0.00	57.91	60.25
11	27.08	28.08	27.86	3.12	27.89	29.91	29.09
12	52.08	54.29	49.95	4.68	51.92	52.73	56.11
13	73.19	69.03	47.27	12.73	0.00	66.65	65.86
14	69.14	69.88	48.26	18.60	68.15	66.32	67.50
15	8.79	8.05	23.71	7.27	0.00	39.28	40.25
16	52.08	54.29	49.95	4.68	51.92	52.73	56.11
17	58.85	58.95	55.50	2.73	59.14	58.85	58.95

Let F be a matrix containing the data units of a feature in its cells, with $m = 8$ rows representing day sections and $n = 7$ columns representing weekdays. The cell containing the most data units is referred to as the baseline cell, and $max(F)$ denotes the number of units in the baseline cell. The average degree of completeness of a feature is computed as follows:

$$degcomp(F) = \frac{\left(\sum_{i=1}^{m} \sum_{j=1}^{n} 100 * \frac{|F_{ij}|}{max(F)}\right) - 100}{(m * n) - 1} \tag{5.3}$$

To exclude the baseline cell from the average degree of completeness, 100 is subtracted and the result divided by the number of cells minus one (the baseline cell). In addition to the degree of completeness, the average number of days per subject and feature is presented in Table 5.14.

Table 5.14.: Average days per subject and feature (A=Acceleration, M=Magnetic Field, Loc=Location, N=Noise, Lig=Ambient Light, B=Battery, W=WiFi)

Subject	**A**	**M**	**Loc**	**N**	**Lig**	**B**	**W**
1	11.46	12.07	8.29	0.67	10.42	12.35	12.57
2	10.12	10.08	5.17	1.68	10.03	12.84	12.79
3	4.06	4.02	3.74	0.02	NA	4.01	3.96
4	6.25	6.27	5.13	0.36	6.28	6.45	6.43
5	33.60	33.57	25.15	1.08	33.96	34.74	34.54
6	11.15	10.98	10.00	0.42	11.27	11.31	10.85
7	15.64	13.15	15.31	0.07	NA	15.92	15.71
8	12.07	11.93	8.63	0.29	12.26	12.36	11.99
9	14.21	13.96	12.49	0.17	14.24	14.74	14.32
10	7.52	7.31	6.27	0.17	NA	7.82	7.52
11	3.60	3.52	3.60	0.34	3.70	4.15	3.85
12	9.35	9.18	8.98	0.30	9.50	9.64	9.67
13	17.19	14.85	10.77	0.48	NA	18.83	18.83
14	17.89	17.84	12.95	1.74	18.10	18.29	18.38
15	0.21	0.23	2.09	0.03	NA	5.11	5.10
16	9.35	9.18	8.98	0.30	9.50	9.64	9.67
17	3.77	3.78	3.57	0.03	3.79	3.77	3.78

It is evident, that not enough data of the noise feature was collected for all participants. The reason for this error could not be found. Furthermore, some of the devices did not deliver ambient light values. Affected devices are the Samsung Galaxy Ace (GT-S5830(i)) and the Sony Xperia Ray (ST18i). The Galaxy Ace does only include a proximity sensor, so the ambient light feature cannot be collected on these devices. According to the technical specification of the manufacturer, the Xperia Ray does include an ambient light sensor, so the missing data must be related to a bug in the operating system.

An almost complete data set of the acceleration, magnetic field, battery and WiFi features was collected for each subject. However, almost one half of the participants did not deliver enough data and is not considered in the evaluation of the system.

Since the collected data shall be used to build a model of a subject, the next chapter introduces a classification technique referred to as the Support Vector Machine. It lays the foundation for the evaluation of the system which is presented in chapter 7.

6. SVM-based Classification

A Support Vector Machine (SVM) is a supervised learning algorithm for classification tasks, that can also be used for regression and novelty detection purposes. SVMs were first introduced by Vladimir Vapnik in 1982 [Vap82], and have been successfully applied in biometrics, e.g. for the classification of subjects based on their gait [NBB11], the fusion of fingerprint and voiceprint [WLL+09], multimodal face and ear recognition [YFEE10], speaker recognition [FLBG10], and keystroke dynamics [LZC+11]. Since the context-aware system presented in this thesis performs a classification, only this use case of an SVM is considered in this section.

SVMs are binary classifiers, that compute a separating hyperplane with a maximum margin to the training examples. The margin is represented by the perpendicular distance between the separating hyperplane and the training example.

A discrimination between the following three cases is necessary to describe the concept of SVMs for classification purposes:

1. Linear Separability
2. Non-Linear Separability
3. Linear Inseparability

A detailed description for each of these cases is given in this chapter, as well as information on class probabilities and general advantages and disadvantages of using SVMs. The examples presented in this chapter are based on the lecture notes of the machine learning course at the Stanford university [Ng13].

The next subsection provides an introduction to the idea of margins.

6.1. Margins

SVMs compute a separating hyperplane to discriminate between two classes. If a training example is far away from the hyperplane, the confidence in its association with the respective class is high. When a training example is close to the hyperplane, the confidence that it belongs to the class is lower, since a small change to the function that determines the hyperplane would lead to a different classification. An example for three possible separating hyperplanes is depicted in Figure 6.1.

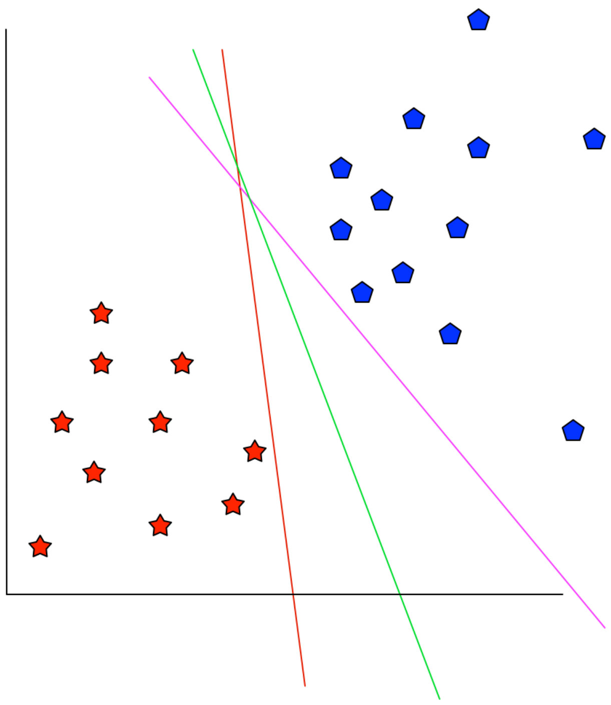


Figure 6.1.: Examples for separating hyperplanes

In order to maintain a high confidence in the classification results, a large margin shall exist between the hyperplane and the closest training examples. Maximising this margin increases the confidence in the classification results.

More formally, two types of margins can be defined. Before discussing these margins, a notation for a linear classifier with class labels y and features x is required. Since SVMs represent binary classifiers, $y \in \{-1, 1\}$ denotes the class labels. The parameters for the classifier are referred to as w and b, so that the hypothesis function can be defined as

$$h_{w,b}(x) = g(w^T x + b) \tag{6.1}$$

with $g(w^T x+b) = 1$ if $w^T x+b \geq 0$, and $g(w^T x+b) = -1$ otherwise. This functionality is provided by the *sign* function.

With this notation, the functional margin $\psi^{(i)}$ of (w, b) for a training example $(x^{(i)}, y^{(i)})$ is defined as

$$\psi^{(i)} = y^{(i)}(w^T x^{(i)} + b) \tag{6.2}$$

Since a large margin represents a high confidence in the classification results, $w^T x + b$ needs to be large positive number for $y = 1$ and a large negative number for $y = -1$. A classification result is correct, whenever $\psi^{(i)} > 0$, since a misclassification always results in a negative number for $\psi^{(i)}$, while $\psi^{(i)} = 0$ means that the example lies on the separating hyperplane. Furthermore, the smallest functional margin of all training examples is used as the functional margin of (w, b):

$$\psi = \min_{i=1,\ldots,m} \psi^{(i)} \tag{6.3}$$

This definition is problematic, since scaling the parameters (w, b) does not change the result of the hypothesis function $h_{w,b}(x)$, but increases ψ. Hence, this is not a good metric for the confidence in a classification result.

A better approach is the geometric margin, which is defined as

$$\gamma^{(i)} = y^{(i)} \left(\left(\frac{w}{\|w\|} \right)^T x^{(i)} + \frac{b}{\|w\|} \right) \tag{6.4}$$

This definition of a margin is invariant with respect to a scaling of (w, b), since the parameters are normalised by dividing the 2-norm of vector w. Again, the smallest geometric margin of all training examples is used as the overall geometric margin of the classifier:

$$\gamma = \min_{i=1,\ldots,m} \gamma^{(i)} \tag{6.5}$$

The training examples with the smallest geometric margin are called the support vectors.

Invariance with respect to scaling the parameters of the classifier allows for a definition of arbitrary constraints, e.g. $\|w\| = 1$, since these can be satisfied by rescaling (w, b). With this constraint, the geometric margin equals the functional margin.

6.2. Linear Separability

Linear separability is given, when the positive and negative training examples can be directly separated by a hyperplane. An example for linear separability and the geometric margin is depicted in Figure 6.2.

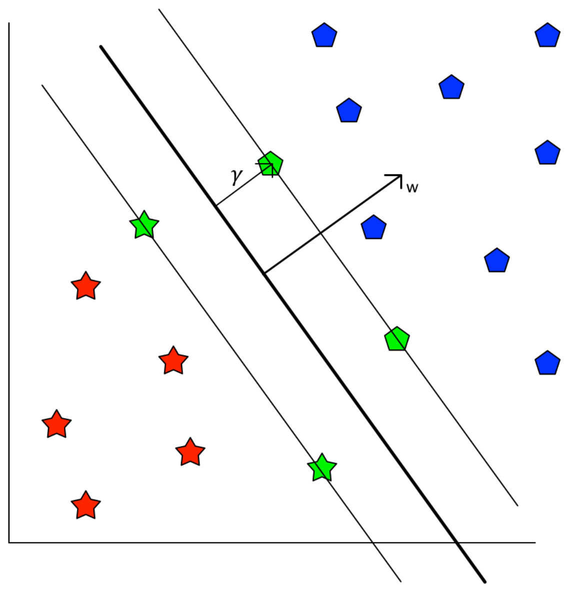


Figure 6.2.: Example for linear separability. The support vectors are indicated by the green colour, w is the parameter vector, and γ the geometric margin

The goal is to maximise the geometric margin γ. Instead of directly maximising the geometric margin γ, which would require constraints that lead to a non-convex optimisation problem, the property of the geometric margin being invariant to scaling (w, b) is used to find a different solution. Scaling (w, b) also scales the functional margin ψ, so the scaling constraint $\psi = 1$ can be imposed. Since the functional and geometric margin are related by $\gamma = \psi / \|w\|$, and taking the scaling constraint into account, $1/\|w\|$ can be maximised instead of γ. Furthermore, maximising $1/\|w\|$ is equivalent to minimising $\|w\|^2$, so the latter is used as it can be solved with quadratic programming (QP) techniques.

An optimisation problem for this task, that can be solved with QP software is defined as follows:

$$\begin{gathered}\min_{w,b} \frac{1}{2} \|w\|^2 \\ \text{subject to } y^{(i)}(w^T x^{(i)} + b) \geq 1, \ i = 1, \ldots, m\end{gathered} \tag{6.6}$$

This is a convex optimisation problem that can be efficiently solved by QP algorithms [BC99].

This optimisation problem can be improved further, so that it can be solved by computing inner vector products, which is the key idea for applying the kernel trick. According to the theory of Lagrange duality, optimisation problems can be represented as primal and dual problems. The dual problem can be solved in lieu of the primal problem under certain conditions [BV09]. Lagrange multipliers represent a method for solving constrained optimisation problems. The dual problem to the optimisation problem is defined as follows [Vap99]:

$$\begin{aligned}
&\max\nolimits_{\alpha} W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \left\langle x^{(i)}, x^{(j)} \right\rangle \\
&\text{s.t. } \alpha_i \geq 0,\ i = 1, \ldots, m \\
&\sum_{i=1}^{m} \alpha_i y^{(i)} = 0
\end{aligned} \tag{6.7}$$

The α_i in the equation are referred to as Lagrange multipliers. From the partial derivative of the Lagrangian with respect to w the following equation can be derived:

$$w = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} \tag{6.8}$$

It follows from the Karush-Kuhn-Tucker (KKT) conditions, that the α_i take on values different from zero solely for the support vectors. Thus, only the support vectors contribute to the position of the separating hyperplane. By using the last equation and plugging it into the hypothesis function of the classifier and simplifying the result, the following equation can be derived:

$$\sum_{i=1}^{m} \alpha_i y^{(i)} \left\langle x^{(i)}, x \right\rangle + b \tag{6.9}$$

Since many of the terms of this sum will be zero, only the inner products of the support vectors with the presented example need to be computed in order to perform a classification.

6.3. Nonlinear Separability and Kernels

There are cases, where the data can only be separated by a nonlinear function, instead of a straight line. An example for such a case is depicted in Figure 6.3.

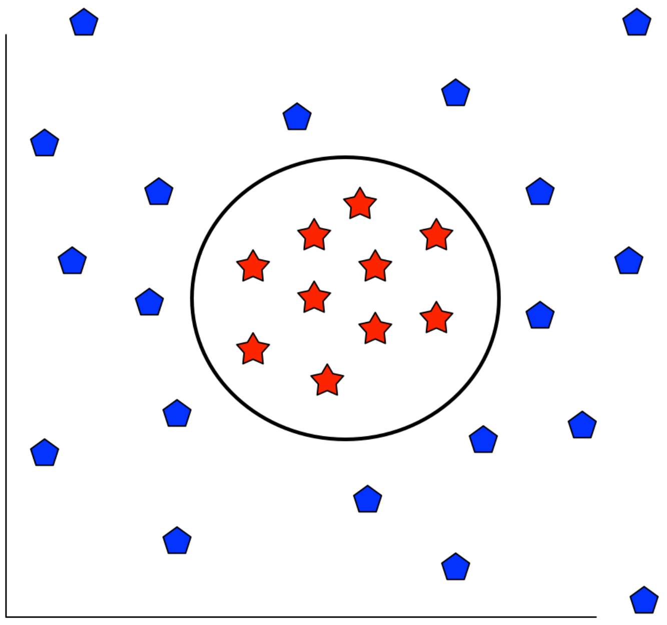


Figure 6.3.: Example for a nonlinear decision boundary

The basic idea is, that the data is mapped to a higher dimensional space, where it is linearly separable. Instead of introducing polynomial terms to the hypothesis to model nonlinear functions, SVMs apply the kernel trick to compute nonlinear decision boundaries.

In order to illustrate the concept of kernels, a simple example is considered [Ng13]. A mapping ϕ of the feature vector x to a higher dimensional space could be performed as follows:

$$\phi(x) = \begin{bmatrix} x \\ x^2 \\ x^3 \end{bmatrix}$$

Instead of training an SVM with the original feature vector, the transformed vector can be used. Replacing $\langle x, z \rangle$ in equation (6.9) with $\langle \phi(x), \phi(z) \rangle$ allows the definition of the following kernel

$$\kappa(x, z) = \phi(x)^T \phi(z)$$

The computation of all inner vector products can now be replaced with $\kappa(x, z)$, which can be efficiently computed, even with a high-dimensional feature mapping $\phi(x)$. A simple example is illustrated by the following polynomial kernel

$$\kappa(x, z) = (x^T z)^2$$

This can be expanded to

$$\begin{aligned}\kappa(x, z) &= \left(\sum_{i=1}^{n} x_i z_i\right)\left(\sum_{j=1}^{n} x_j z_j\right)\\ &= \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j z_i z_j\\ &= \sum_{i,j=1}^{n} (x_i x_j)(z_i z_j)\end{aligned}$$

This computation has a time complexity of $O(n^2)$ and the resulting feature vector for $n = 3$ is given by

$$\phi(x) = \begin{bmatrix} x_1x_1 \\ x_1x_2 \\ x_1x_3 \\ x_2x_1 \\ x_2x_2 \\ x_2x_3 \\ x_3x_1 \\ x_3x_2 \\ x_3x_3 \end{bmatrix}$$

Considering the definition of the Kernel, it can be seen that the computation requires only linear time $O(n)$ compared to the quadratic time complexity for mapping the input features to a high dimensional space. Hence, it is not necessary to compute the expensive feature mapping.

Kernels can also be regarded as a measure of similarity. The inner product of two vectors is large if they are similar, and zero in case they are orthogonal. Thus, functions that compute the similarity between two vectors are candidates for kernel functions. An example is the Gaussian radial basis function kernel (RBF), which is also used for the training of an SVM in the presented thesis

$$\kappa(x, z) = \exp\left(-\gamma \left\|x - z\right\|^2\right) \tag{6.10}$$

The necessary and sufficient conditions for a valid kernel function are given by Mercer's theorem [Mer09], which basically states that the kernel matrix needs to be symmetrical and positive semi-definite.

6.4. Linear Inseparability and Outliers

The concept of an SVM presented so far assumes linear separability of the input data and is sensitive to outliers. Consider the example depicted in Figure 6.4: outliers result in a much smaller geometric margin and unintuitive position of the separating hyperplane, which may result in a reduced classification performance.

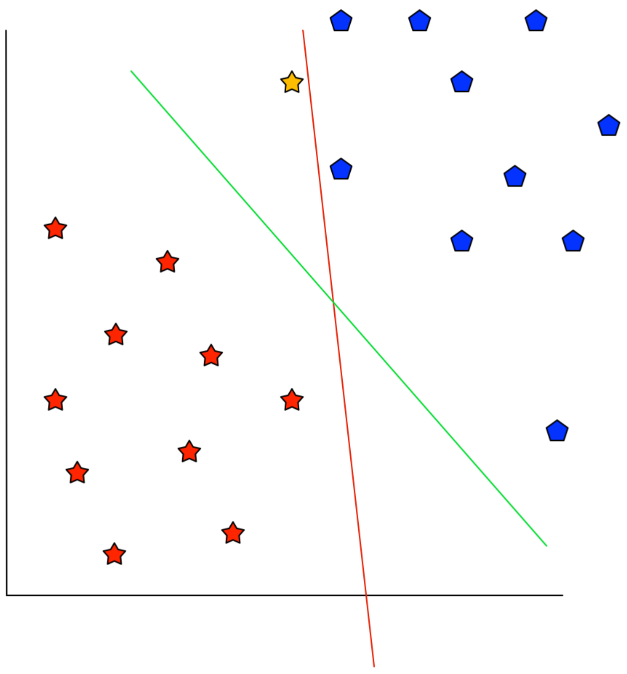


Figure 6.4.: Example for an outlier that changes the position of the separating hyperplane. The green hyperplane is desirable, but the red one is computed due to the presence of the outlier indicated by the orange colour.

Furthermore, the current SVM formulation cannot find a separating hyperplane if the data set is linearly inseparable. While mapping the input data to higher dimensional feature spaces, as described in the last subsection, increases the likelihood of linear separability [Cov65], there is no guarantee for it. An example for a linearly inseparable data set is given in Figure 6.5.

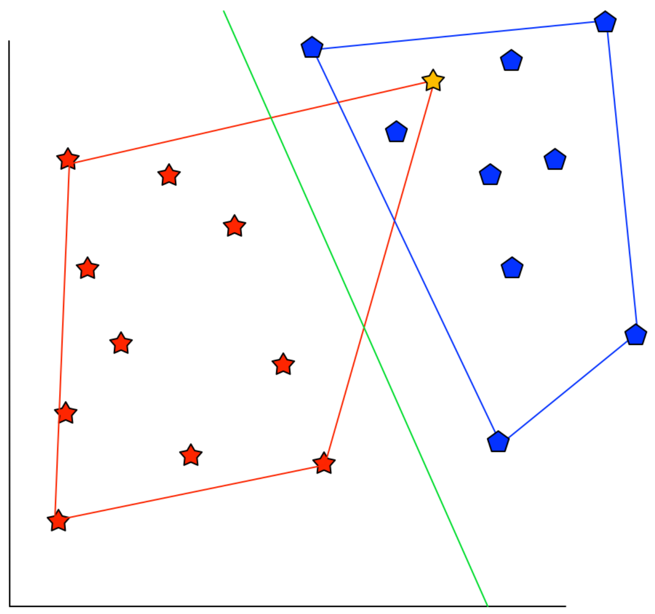


Figure 6.5.: Example for a linearly inseparable data set

Visualising the convex hulls of the training examples for each class makes the linear inseparability obvious. Without the outlier indicated by the orange colour, the decision boundary could be computed as indicated by the green line. In order to solve this problem, the margin constraint of the optimisation problem needs to be relaxed and combined with a regularisation term in the objective function, which leads to the following formulation of the optimisation problem [Bur98]

$$\begin{gathered}
\min_{w,b} \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i \\
\text{s.t. } y^{(i)}(w^T x^{(i)} + b) \geq 1 - \xi_i,\ i = 1, \ldots, m \\
\xi_i \geq 0,\ i = 1, \ldots, m
\end{gathered} \tag{6.11}$$

The modified constraint allows training examples to be within the margin or on the wrong side of the separating hyperplane. Specifically, it allows a functional margin that is smaller than one. However, the regularisation term provides a balance to the relaxed margin criterion, so that the objective function is optimised for small ξ_i and a large margin.

When the cost parameter C goes to infinity, the resulting hyperplane is similar to the one computed without regularisation and relaxed margins. However, the value of C has to be determined empirically.

Again, this problem can be represented as a primal and dual problem. The dual problem differs only slightly from the original problem:

$$\begin{aligned} &\max\nolimits_\alpha W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j \left\langle x^{(i)}, x^{(j)} \right\rangle \\ &\text{s.t. } 0 \leq \alpha_i \leq C, \ i = 1, \dots, m \\ &\sum_{i=1}^{m} \alpha_i y^{(i)} = 0 \end{aligned} \tag{6.12}$$

The only change introduced to the original formulation is $\alpha \leq C$. With this result, the description of the SVM as used in the presented thesis is complete.

6.5. Class Probabilities

Since the output of an SVM is a label $y \in \{-1, 1\}$ that denotes the class membership, probability estimates must be computed separately for the classes. A well-known method to derive such probabilities is logistic regression. Logistic regression employs a sigmoid function to map input values to a class. An example for such a sigmoid function is

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

This function maps to $y = 1$ for $\theta^T x > 0$ and $y = -1$ otherwise. A big absolute value of $\theta^T x$ corresponds to a high confidence in the classification result. The sigmoid function is depicted in Figure 6.6.

The confidence in a classification result can be modelled by introducing a scaling factor for $\theta^T x$.

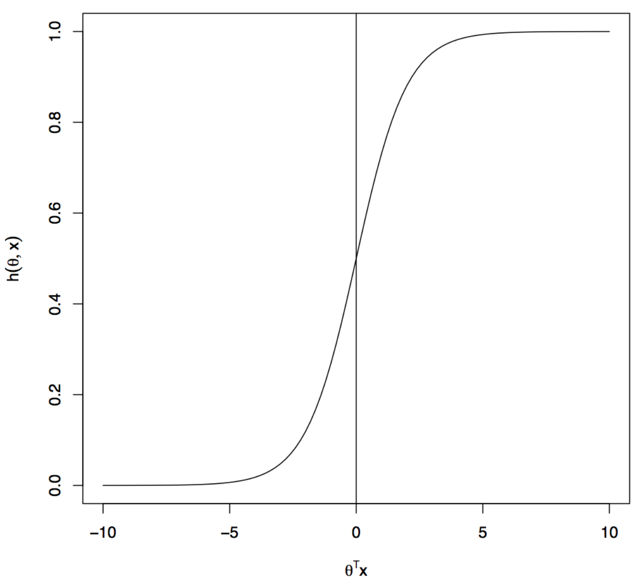


Figure 6.6.: Example for a sigmoid function. The decision boundary is at $\theta^T x = 0$. The estimated probability for the positive class is given by $h_\theta(x)$, and for the negative class by $1 - h_\theta(x)$.

6.6. Advantages and Disadvantages

One of the advantages of the SVM is the availability of efficient optimisation algorithms, which solve the underlying quadratic optimisation problem. In contrast to other machine learning techniques such as artificial neural networks or decision trees, SVMs always find the global optimum due to the underlying convex optimisation problem.

Only a small number of parameters need to be fine-tuned SVMs. In the case of the RBF kernel, only the kernel parameter γ and the regularisation parameter C need to be selected.

By applying the kernel trick, the probability of linear separability is increased without increasing the computational complexity.

However, a problem that is not addressed by SVMs is the feature selection. The selection process still requires the domain-knowledge of experts in the respective field. Additionally, correlated features cannot be transformed automatically, so that a manual analysis is required to do so.

SVMs require numerical attributes in order to train a model. For instance, categorial attributes need to be transformed into numerical values before they can be used in a feature vector. Furthermore, it is considered a best practice to perform feature scaling before training a model. Features can be scaled, e.g. by subtracting the mean and dividing by the standard deviation of the underlying distribution.

In the next chapter, an SVM is trained with the collected data and its performance is evaluated.

7. Subject Modelling

In this chapter, the process for building a subject-specific model based on the collected data is outlined. Context data frames are used to train an SVM for each of the participants. The resulting model is evaluated on a test set and the results are discussed. In addition, a specific subject is selected based on the performance of the SVM to analyse the contextual data.

7.1. Related Work

Using contextual data in order to authenticate subjects is also referred to as implicit authentication. For instance, Shi et al. proposed a system that uses phone, SMS, GPS, and browser data to implicitly authenticate subjects [SNJC11]. Their approach is based on the discrimination between good and bad events, that increase or lower a confidence score. An active authentication is required as soon as the score falls below a certain threshold. A subject model is built by determining the probability density functions for each feature, which are conditioned on the weekday and time of day. Their results show, that an adversary is locked out of the system after 16 usages of the device with 95% probability and two usages with 50% probability. The most interesting feature in the evaluations turned out to be the GPS information. The best performance could be achieved by combining all the aforementioned features.

Hulsebosch et al. [HBL$^+$07] estimate the probability for a subjects' current location in order to perform such implicit authentication utilising multiple location sensors. Multiple distributed sensors are required in their approach, i.e. they must be spread across a building. Again, the probabilities of multiple sensors are fused in order to compute the total probability of a subject being in a certain location at a certain time.

A work that is similar to the approach in the presented thesis was published by Lima et al. [LRAD11]. They collect environmental information as part of a spatial-temporal context, as well as behavioural information in order to implicitly authenticate a subject, or to determine whether the subject shall be challenged or not. Their approach is based on the clustering of contextual data and performing anomaly detection. A new event is associated with a genuine subject, if it is within the standard deviation of the underlying distribution. However, they did not perform an evaluation of the proposed system, so that its performance cannot be compared to aforementioned approaches.

7.2. Context Modelling

Before a subject-specific model can be built, a context model is required. Such a model is comprised of contextual information, which describes a distinct situation. Contextual information is obtained from sensors, preprocessed, and aggregated into a context data frame (see Figure 4.2). Since the system is designed as a mobile application, data acquired from all available sensors, as well as information related to user-smartphone-interaction is included.

Since varying amounts of feature instances were collected per weekday and day section, as indicated by the average degree of completeness in Table 5.13, the question arises how missing feature instances shall be treated. In many cases, more data of a specific feature was collected, thus the remaining features have to be filled with defined values as part of the aggregation phase. A common approach is to replace missing data by the mean, so that the underlying distribution does not change. However, this would add incorrect contextual information. For instance, latitude and longitude pairs of locations a subject visits are collected. Inserting the mean for missing data would add locations to the training set, which the subject never visited but that may be close to locations an impostor visits. Thus, missing values are set to zero. However, this represents a problem for existing binary features, which take on values $x \in \{0, 1\}$. This issue is solved by increasing the measured value by one, so that after the transformation $t(x) = x + 1$ the features take on values $x_t \in \{0, 1, 2\}$, with zero denoting a missing value, one denoting a logical false, and two a logical true.

All collected features are grouped into a matrix as described in subsection 5.7.3, so they can be matched by their cell and fused into a single vector, which is referred to as a context data frame. Each context data frame contains an instance of each feature. Each cell of the matrix denotes a day section of a three hour duration. Since the data is collected in 20 minute intervals, this leads to three feature measurements per hour.

An issue with the combination of features is, that while the data units in the feature matrix are mapped to a certain day section, it is not defined at which of the 20 minute intervals they were collected. This can lead to a situation were features are combined, that have a temporal difference of 40 minutes. Combining the features based on their part of a day section would lead to numerous missing values, since the individual sensor activations are not synchronised. For instance, feature A might have been collected at minute 20, and feature B at minute 21. While both features are part of the same measurement, they would be mapped to different parts of a day section. So the fusion process may create inaccurate context descriptions, that contain a mix of temporally unrelated feature instances, but this is a limitation that is accepted due to the coarse-grained temporal resolution of the current implementation. A fine-grained temporal resolution requires smaller data collection intervals, which would even further increase the power consumption of the application. This trade-off might become a non-issue with better battery technology or more energy-efficient sensors and CPUs. A non-scalable solution to this issue is to first compare the date and then compute a diff of the timestamps between all features and combine those that do not exceed a threshold of e.g. five minutes. A solution for a production system is to combine the features in the context component of the context subsystem and store the context data frame instead of the individual features. Still, the question of how to integrate the non-periodically collected features remains. These methods represent improvements over the presented approach, but are not implemented due to the time constraints of the presented thesis, and are subject to future work. The resulting context data frame is a 19-dimensional vector, as depicted in Table 7.1.

The WiFi feature was removed, since a transformation of the hash to a numerical value leads to very large numbers. The result would have to be scaled down to a much smaller value as part of the feature scaling and thus, lose precision, which might increase the probability of collisions. Thus, including the WiFi feature in the evaluation is subject to future work. These vectors are used for the training of an SVM in order to compute a subject-specific model. The next subsection explains the training process in detail.

7.3. Context-based Subject Modelling

While the context model is defined by a composition of situational, environmental, and behavioural information, a subject-specific model has to be derived from a set of context data frames. The goal of building a subject-specific model is to perform a classification of context data frames in order to determine if the current context is associated with the enrolled subject. Solutions to classification problems are provided by machine learning methods. An SVM in combination with logistic regression for class probabilities is used in the presented thesis.

Table 7.1.: 19-dimensional feature vector for the training of an SVM

Position	**Feature**
1	Weekday
2	Day section
3	Acceleration x
4	Acceleration y
5	Acceleration z
6	Magnetic Field x
7	Magnetic Field y
8	Magnetic Field z
9	Battery Percentage
10	Battery Charge Indicator
11	Illuminance
12	Latitude
13	Longitude
14	Location Accuracy
15	Duration of Screen Activity
16	Boot Indicator
17	Call Duration
18	Call Direction
19	Number in Contact List Indicator

A predefined set containing negative examples is labeled with class –1, while the subjects' context data frames are labeled with class 1. A simple cutting strategy is applied by dividing the complete set of context data frames for each of the classes into a training set of size 4/5 and a test set of size 1/5, and combining the respective sets. The employed libsvm[1] implementation of a support vector machine performs cross-validation on the training data, i.e. a cross-validation set is not part of the aforementioned cutting strategy.

Since the data set is ordered by weekday and day section, it cannot be cut directly at the aforementioned boundaries, since the sets would include disjoint temporal information. The following strategies are applied for the creation of training and test sets:

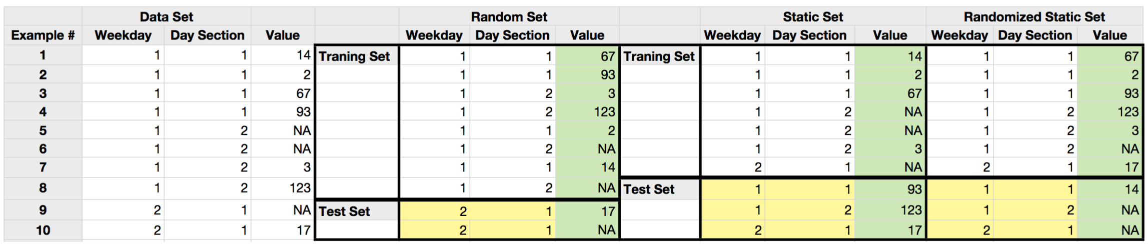
Random Sets A simple approach to create training and test sets is to apply a random permutation on the complete sets for the classes, thus assigning random context data frames to the training and test sets. This method may lead to the problem, that examples for a certain weekday and day section combination are only assigned to one of the sets, thus decreasing the classification performance.

Static Sets Instead of applying random permutations for the assignment of context data frames to the training and test sets, the data set can be divided by class, weekday, and day section. The aforementioned cutting strategy is then applied to the lowest level represented by the day sections. This method ensures, that most weekday and day section combinations are represented in both the training and the test set. Still, if there are more combinations of weekday and day section than slots in the test set, some combinations will not be represented in the test set. This is not an issue, since it is more important, that a good representation of context data frames is included in the training set. An example for this is depicted in Figure 7.1.

Randomised Static Sets Using static sets instead of random sets solves the issue of balancing the assignment of context data frames to the respective sets. However, not all context data frames might include discriminative information (e.g. if the frames include a lot of missing values), which can lead to discriminative context data frames being assigned to the test set only, which would also decrease the classification performance. This issue can be solved by applying a random permutation on the lowest level, which is represented by the day sections. By randomising

[1]`http://www.csie.ntu.edu.tw/~cjlin/libsvm/`

the static sets and performing multiple evaluations, the data set can be explored, and the likelihood of including discriminative data in the training and test sets increases.



	Data Set				Random Set				Static Set			Randomized Static Set		
Example #	Weekday	Day Section	Value		Weekday	Day Section	Value		Weekday	Day Section	Value	Weekday	Day Section	Value
1	1	1	14	Traning Set	1	1	67	Traning Set	1	1	14	1	1	67
2	1	1	2		1	1	93		1	1	2	1	1	2
3	1	1	67		1	2	3		1	1	67	1	1	93
4	1	1	93		1	2	123		1	2	NA	1	2	123
5	1	2	NA		1	1	2		1	2	NA	1	2	3
6	1	2	NA		1	2	NA		1	2	3	1	2	NA
7	1	2	3		1	1	14		2	1	NA	2	1	17
8	1	2	123		1	2	NA	Test Set	1	1	93	1	1	14
9	2	1	NA	Test Set	2	1	17		1	2	123	1	2	NA
10	2	1	17		2	1	NA		2	1	17	2	1	NA

Figure 7.1.: Examples for the creation of training and test sets. The random set strategy assigned examples to the test set, that are not represented in the training set. The randomised static set has a different order of the day sections compared to the static set. The different training and test set sizes between the random and static set variants are indicated by the yellow coloured background.

Randomised static sets are used for the training of the classifier in the presented thesis, since these have a higher likelihood of distributing good context representations among the training and test sets. It should be noted, that results can only be reproduced by storing a snapshot of the best performing partition of the data set, since the random permutation creates different training and test sets every time. In addition, this strategy leads to different total sizes of the training and test sets. For instance, in the example from Figure 7.1 the complete data set would normally be divided into a training set of size eight and a test set of size two. By applying the cutting strategy to the day sections, the test set size increases. This effect is desirable, since it represents a better distribution of relevant context data frames.

Furthermore, it is considered a good practice to remove redundancy from the data sets. Aside from removing duplicate entries, this can be done by searching for strongly correlated features and investigate if one of these can be removed. This preprocessing step is not performed due to time constraints of the presented thesis and is subject to future work.

In the next subsection, an evaluation of the proposed system is presented.

7.4. Evaluation

This subsection provides an overview of the classification performance and learning curves for different training and test set sizes. Furthermore, the contribution of individual features is determined by analysing the weights of the parameter vector that is computed by the SVM.

The radial basis function (RBF) kernel is employed for the training, with the parameters $\gamma = 0.1$ for the kernel function and $C = 10$ for the cost. The initial parameters are chosen arbitrarily and are subject to change with respect to tuning capability of the e1071 package[2] of the R programming language, which performs a grid search on defined intervals for the parameters. In addition to the decision output values of the SVM, the respective class probabilities are computed. The required functionality, which is based on a logistic regression like sigmoid function (see section 6.5), is provided by the libsvm.

7.4.1. Classification Performance

In order to evaluate the classification performance, the precision P, recall R and the resulting F_1-scores are computed. Let T_p , F_p , F_n denote the amount of true-positives, false- positives, and false-negatives, then the applied metrics are defined as:

$$P = \frac{T_p}{T_p + F_p}\ ,\ R = \frac{T_p}{T_p + F_n}\ ,\ F_1 = 2\frac{PR}{P + R}. \tag{7.1}$$

Accuracy is not considered as a performance measure, because of the unbalanced classes in the training set (there are a lot more examples for the negative class).

The results for an arbitrary evaluation are given in table Table 7.2. Five of the generated models have F_1-scores of over 70%, all except two exhibit a precision of more than 70%, and four a precision of over 90% with varying recall scores. A good or bad F_1-score does not seem to be related to the amount of positive class training examples or average degree of completeness (see Table 5.13), so other factors must be influencing the quality of the model. Varying the decision threshold for the positive class probability leads to better precision or recall scores, but cannot improve the F_1-score.

Performing the evaluation again leads to slightly different results, which is due to the fact, that the data for positive and negative examples is permuted for individual day sections before being assigned to training and test sets. A permutation is employed in order to explore the data set, so that missing values or less discriminative examples do not

[2]`http://cran.r-project.org/web/packages/e1071/index.html`

Table 7.2.: Classification performance for all subjects with the parameters $\gamma = 0.1$ and $C = 10$. The results are computed based on the confusion matrices. The set size denotes the combined number of positive class examples, which are distributed among the training and test set. The data for models 12 and 16 contains numerous undefined values, thus the missing performance values.

ID	**Precision**	**Recall**	F_1**-Score**	**Set Size**
1	0.85	0.32	0.46	2364
2	0.78	0.43	0.56	2268
3	0.74	0.15	0.25	865
4	0.82	0.42	0.56	383
5	0.95	0.82	0.88	5536
6	0.97	0.84	0.9	2023
7	0.95	0.86	0.9	2782
8	0.76	0.21	0.33	2440
9	0.77	0.34	0.47	3190
10	0.78	0.18	0.3	1333
11	0.78	0.18	0.3	1106
12	0	0	NaN	1842
13	0.83	0.43	0.56	3349
14	0.87	0.62	0.72	3938
15	0.8	0.41	0.55	1099
16	0	0	NaN	1842
17	0.92	0.76	0.83	715

always end up in the training set. The varying performance of the models indicates, that some of the training examples carry more information than others. An analysis of the relevant features for good and bad performing models is provided in a later subsection.

It has to be considered, that the results are based on a very large number of training examples per subject. The duration of the computation for a single model ranges from two to twelve minutes, which is not feasible on mobile devices. Thus, the next subsection provides an evaluation for the performance of a good model for different training set sizes.

Another aspect is, that no advanced adversary model is employed in order to evaluate the performance of the system. For instance, most of the examples have spatially disjoint location information. In order to model an adversary, the location of a genuine subject and impostor needs to be identical at some point in time. Shi et al. [SNJC11] employ a splicing strategy, which is based on computing a delta of the adversaries location and the location of the genuine subject at a defined point in time. This delta is added subsequently to the other locations of the adversary in this particular time series. This strategy does not make sense for the presented context model, since the information is not recorded as a fine-grained time series, but rather as part of a day section. Again, this is a matter of granularity or temporal resolution, that can be addressed in future work.

The classification results indicate, that a discrimination between subjects based on contextual information is feasible.

7.4.2. Performance Curve

The presented classification results are based on a training set size of approximately 28.000 vectors, and a test set size of approximately 7.000 vectors. Hence, the computation of a model takes a significant amount of time, which is not feasible on a resource constraint mobile device.

Since the overwhelming majority of training examples belong to the negative class, which is comprised of examples for all other subjects, the first step is to reduce the amount of these. Including twice the amount of negative training examples compared to positive examples is a heuristic that is applied for this task. Furthermore, in order to better understand how much training examples are required, the amount of training examples is gradually increased. The test set size remains constant, since the goal is to measure the generalisation performance of the model relative to the size of the training set.

The models with the id 5, 6, 7, 14, and 17 have a good performance on the test data, as denoted by the respective F_1-score (see Table 7.2). Since the model with id 5 has the most positive class examples, it is used for this evaluation.

Ten gradually increasing training set sizes are used to train and evaluate the model. In each step, a part of the data for each weekday and day section combination is included in the training set for the positive class. The number of training examples $T(s, d)$ for each step $s \in \{1, 2, \ldots, 10\}$ and current number of examples for the day section d is computed as denoted by the following simple formula:

$$T(s, d) = s * \frac{1}{10} * d$$

After the number of positive class examples for the current iteration is computed, the number of negative class examples is determined by multiplying this number by two. When all weekday and day section parts are created, these are combined into a single training set data frame, which is used to train the model.

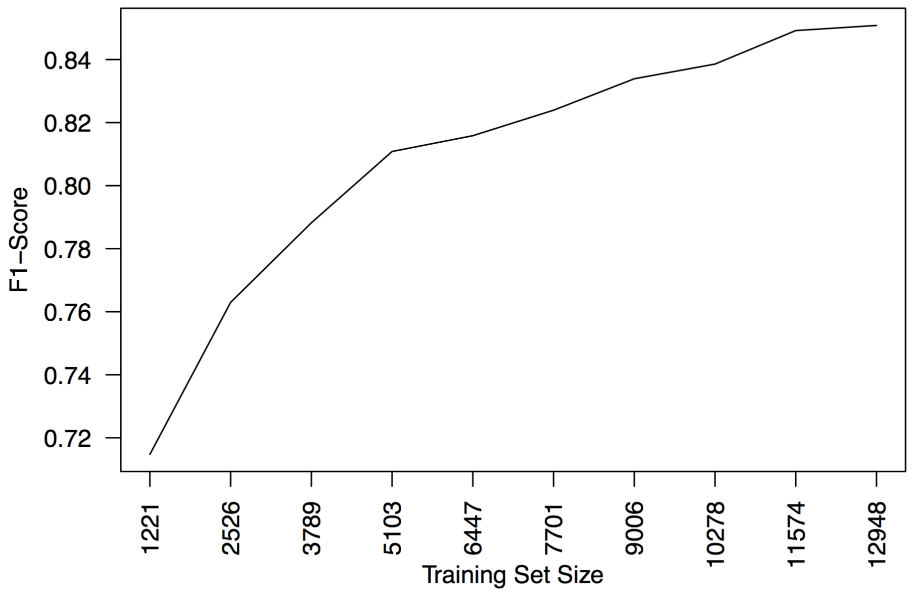


Figure 7.2.: F_1-scores for the model with id 5 for gradually increasing training set sizes. The set size is comprised of positive and negative examples, in contrast to the set sizes given in Table 7.2, and must be compared with the original training set size of approximately 28.000.

An F_1-score of over 80% can already be achieved with a training set of size 5.000, as indicated by Figure 7.2. This amounts to an 8% loss of performance for a reduction of the training set size by a factor > 5, or a 4% loss of performance for a reduction by a factor > 2.

This result indicates, that a far smaller training set can be used without sacrificing too much performance. Since the required time for training the model is a crucial factor for the computation on mobile devices, a benchmark for the different training set sizes is performed. The results are depicted in Figure 7.3.

It can be seen, that reducing the original training set by a factor of 2.16 also reduces the time required for the training of the SVM by a factor of 3.34, while still maintaining an F_1-score > 0.84. This benchmark was performed on a quad-core Intel Core i7 with 2.3GHz and 8GB of RAM. Performing these operations on a resource-constraint mobile device would lead to increased durations for the training. Thus, only small training sets are a viable option for the training on mobile devices. However, long running computations could also be performed at night and when the device is attached to a power source.

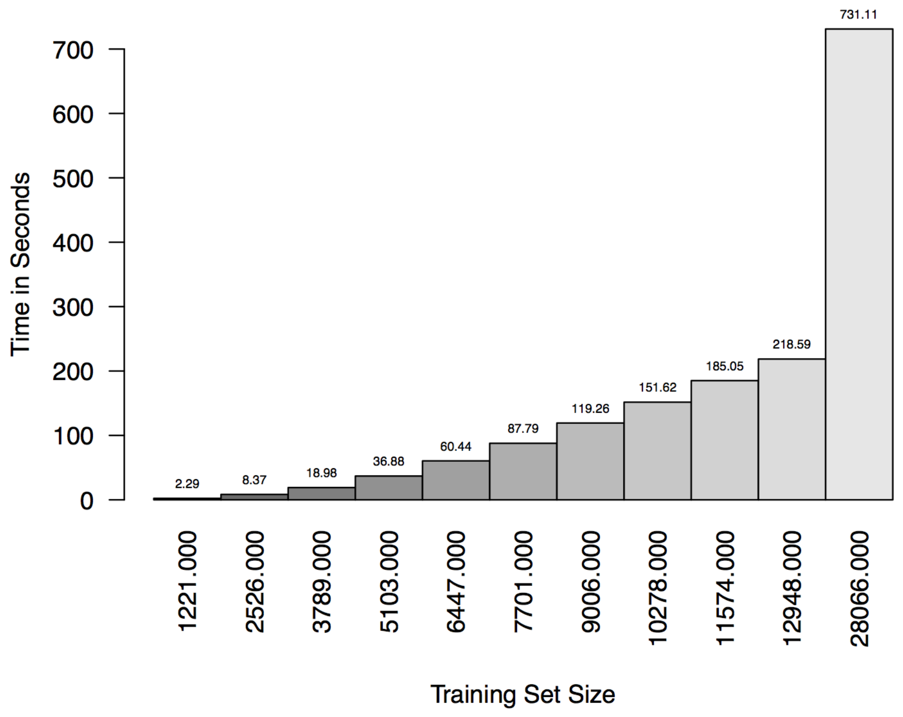


Figure 7.3.: Time in seconds for the training of the SVM for model with id 5 with gradually increasing training set sizes

7.4.3. Contribution of Features

A question that still remains is, which features contribute most to the classification decision. The contribution of each feature is equivalent to its weight in the parameter vector of the SVM, which can be directly retrieved from the computed models. Figure 7.4 depicts an overview of the feature weights of three selected models. In order to gain comparable weight vectors, the absolute value of each weight is computed, and the vectors are normalised.

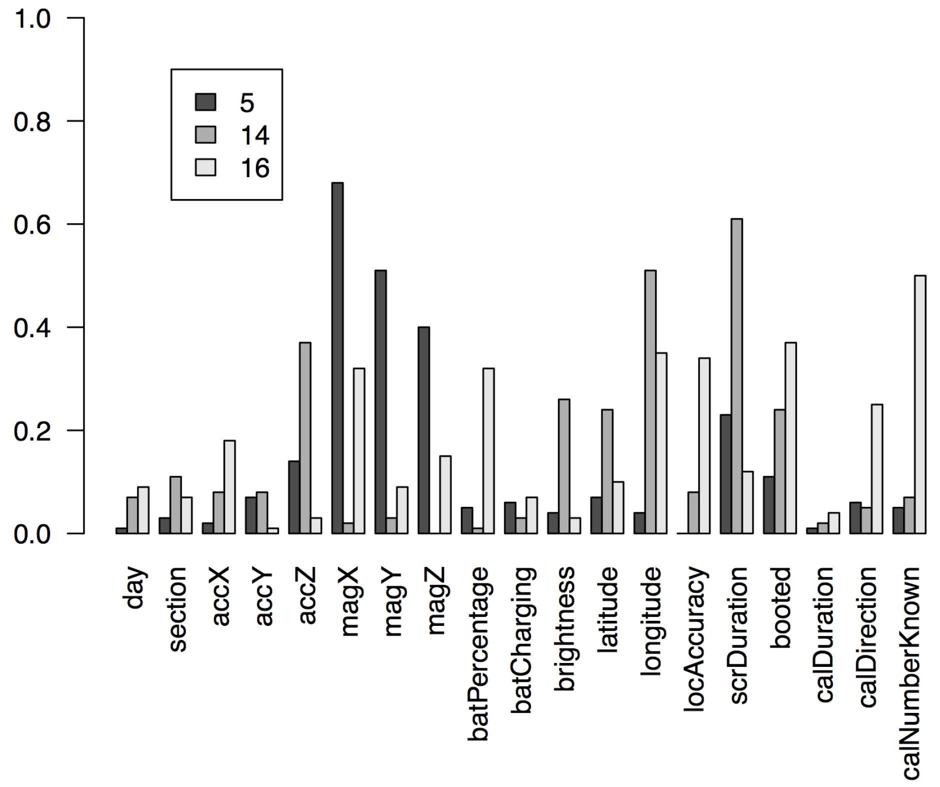


Figure 7.4.: Normalised weights of the individual features for the models 5, 14, 16

The results indicate that the most relevant features vary greatly from subject to subject. Thus, a general feature selection cannot be performed on the basis of the weights computed by the SVM. Instead, it might be more sensible to include even more discriminative features.

An inherent problem of the feature analysis is, that the data was recorded on different devices with potentially different sensor characteristics. Thus, there is a high risk associated with giving a strong weight to a feature that was collected by a physical sensor, because it might emphasise characteristics of the specific device instead of behavioural characteristics of the subject. This issue could be mitigated by calibrating the sensors of all devices to ensure comparable values under controlled conditions. Unfortunately, this is not possible due to the distributed and anonymous nature of the performed data collection. Another solution for this issue is to perform these evaluations on a single device with multiple subjects, which is subject to future work.

However, when building a sum over the normalised feature weights of all models, a trend towards behavioural features becomes visible, as depicted in Figure 7.5.

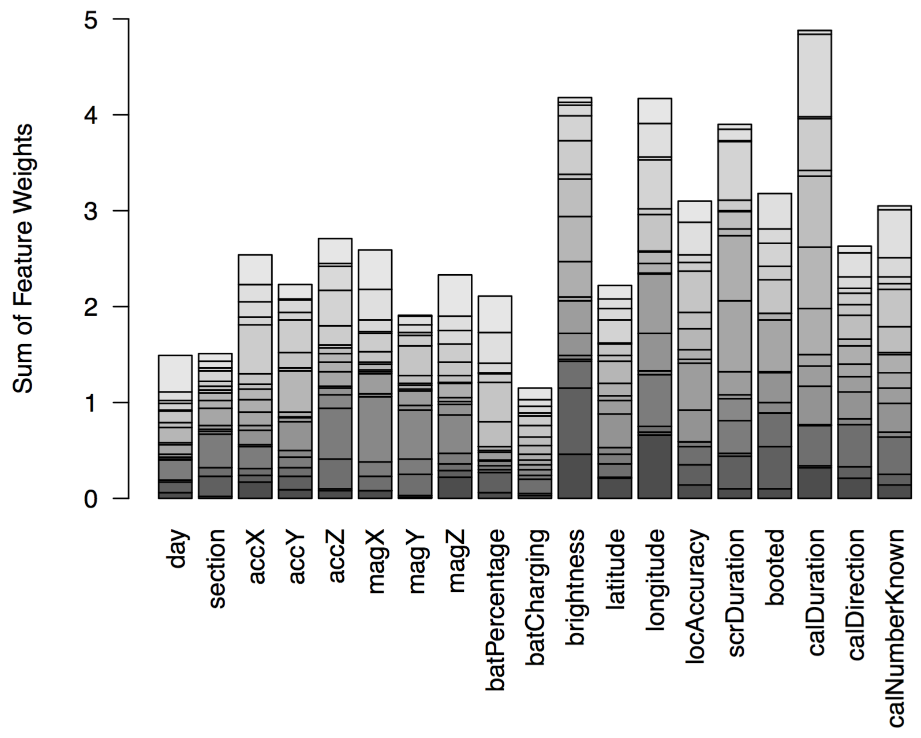


Figure 7.5.: Sum over the normalised feature weights for all models

This result is not yet very relevant, considering that it also contains the feature weights of models with a bad performance. Thus, the average feature weights shall be compared between the models with a good and bad performance. The comparison result is depicted in Figure 7.6. It is obvious, that good models assign a higher weight to the location feature (longitude) and a much smaller weight to calling behaviour related features. It is not surprising that the location feature is a strong discriminator, so the performance of the system decreases significantly in its absence. The high weight for the magnetic field values may be due to sensor characteristics and is biased by the high weight assigned by model with id 5 for this specific feature. An interesting result is, that both good and bad models assign a high weight to the duration of device usage feature (scrDuration), so it might be worthwhile to add more features that model the subjects interaction with the device more precisely.

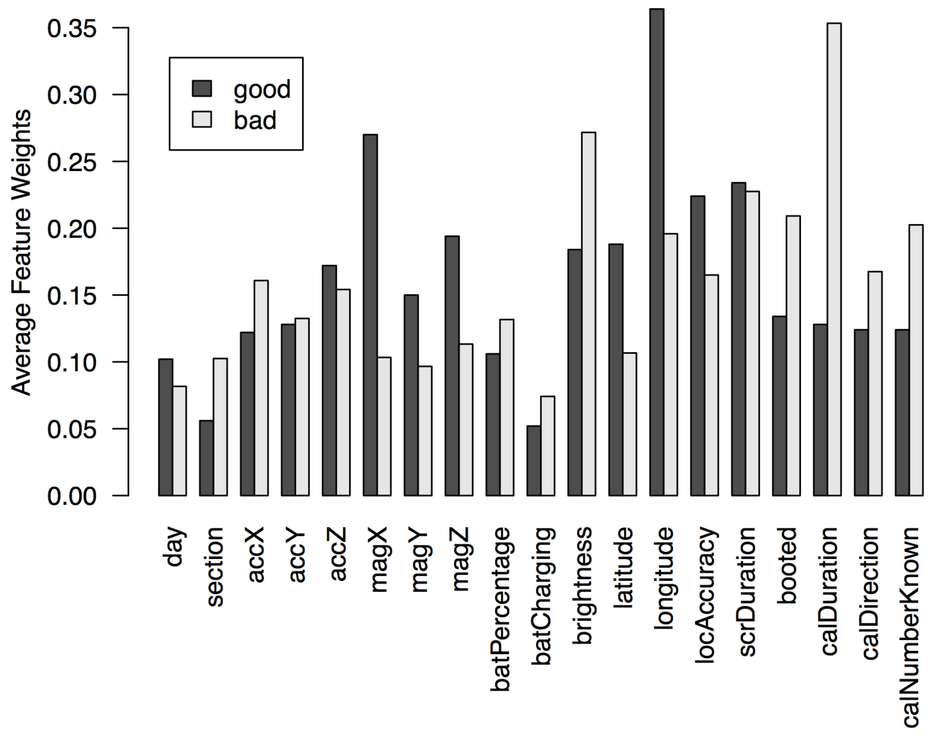


Figure 7.6.: Comparison of average feature weights between the good models (id 5,6,7,14,17) and the rest.

Furthermore, a step that is normally performed before the training of an SVM is to determine the correlation between features, and the transformation or removal of one of the features which are strongly correlated. A correlation diagram that is created using Pearsons' correlation coefficient is depicted in Figure 7.7. A deep colour denotes a strong, blue colour a positive, and red colour a negative correlation. An alternative presentation of this relationship is depicted by the pie charts in the lower panel.

Five feature groups have a medium to strong correlation in the diagram:

- Accelerometer (X,Y,Z)
- Magnetic Field (X,Y,Z)
- Location (Latitude, Longitude, Accuracy)
- Battery percentage and charging indicator

- Call duration, direction and if the number is in the contact list

Thus, these features are candidates for a transformation or removal. Location information could be transformed by applying a grid, so that there is a single value representation that sufficiently describes a location with a defined precision. Furthermore, the accelerometer data may e.g. be replaced by a discrete value indicating a specific activity. Magnetic field data could be used to derive the direction (N,NE,ES,SW,NW) a subject is facing (ignoring possible errors due to the calibration issue described in subsection 5.3.3). Transforming physical sensor data to discrete categorial attributes has the additional benefit, that sensor or device specific characteristics do not have such a strong influence on the performance of the resulting model.

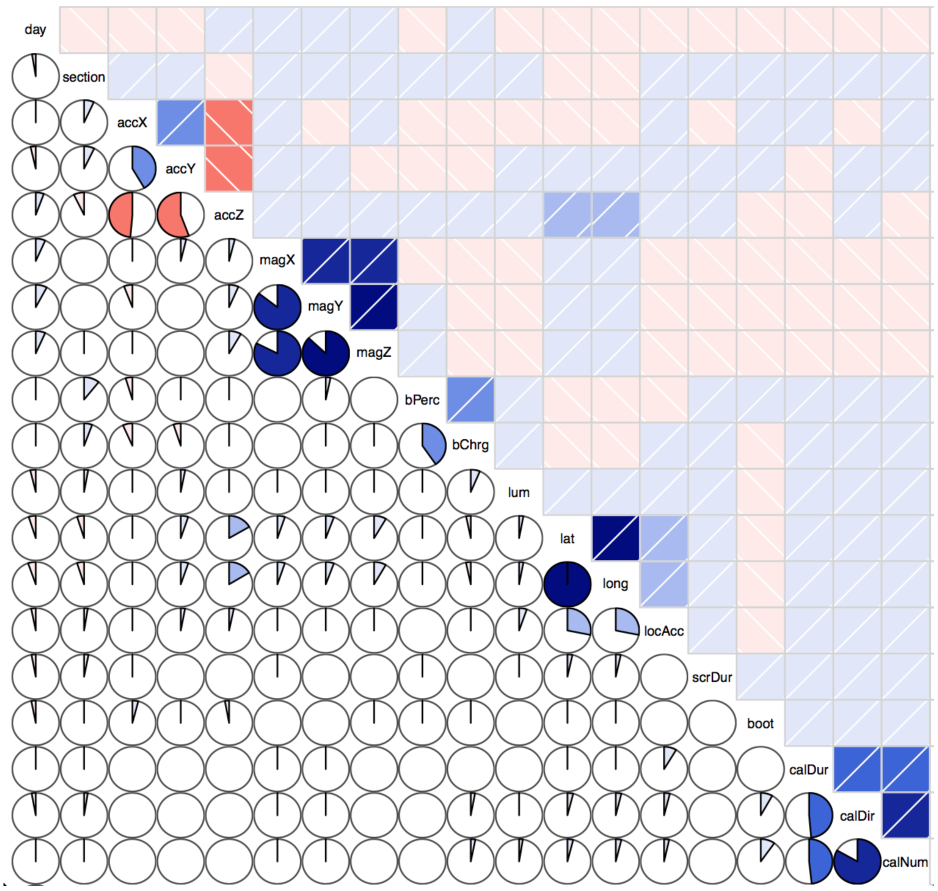


Figure 7.7.: Correlation of all features for the model with id 5. Positive correlation is indicated by the blue colour, negative correlation by the red colour. The colour saturation and pie charts indicate the strength of the correlation.

7.5. Additional Evaluations

In addition to the results presented so far, an evaluation was performed utilising the *KXEN InfiniteInsight*[3] software, which provides methods for classification, regression, and segmentation/clustering. The software performs sophisticated preprocessing and statistical optimisations on data sets prior to the training of a classifier. Optimisations include the binning of continuous variables and compression of the resulting categories based on statistical properties.

The SVM models computed by this software exhibit an even higher performance on the collected data for some models, which is not surprising given its sophisticated preprocessing methods. While the performance of model 5 was improved by 4%, the performance of model 6 was reduced by 14%. However, the performance of bad models was improved significantly. The applied cutting strategy is comprised of periodically choosing three examples for the training set and one for the test set. Custom cutting strategies can be provided, but were omitted for the sake of simplicity. The classification results of this software for a subset of the evaluated models are compared to the evaluation results described in subsection 7.4.1, and are presented in Table 7.3.

Table 7.3.: Performance comparison of some models between KXEN and the approach in the presented thesis (P=precision,R=recall,F_1=F_1-score)

ID	P **KXEN**	R **KXEN**	F_1 **KXEN**	F_1 **thesis**
3	0.59	0.70	0.64	0.25
5	0.91	0.92	0.92	0.88
6	0.86	0.68	0.76	0.9
10	0.81	0.71	0.75	0.3
12	0.5	0.44	0.48	0

The results indicate that improvements are possible, but the model with id 6 as computed in the presented thesis performs better and the performance of model with id 5 improved only slightly. However, the improvements for the bad performing models 3 and 10 were significant. The model with id 12 exhibits a performance almost identical to a random model, which indicates a problem with the underlying data.

The distribution of feature weights for the models computed by this software vary from the results presented in subsection 7.4.3, in that a higher weight is put on the

[3]`http://www.kxen.com/Products`

accelerometer, location, and brightness features. Again, this might be problematic, since the model might adapt to a variance introduced by device specific sensor characteristics.

8. Conclusion

In the presented thesis the design and partial implementation of an architecture for a context-aware mobile biometric system was presented and the constituent components were described in detail. The feasibility of the proposed architecture has been demonstrated by developing a mobile application which has been applied for data collection purposes. Furthermore, the results of the data collection were utilised to train an adequate classifier in order to obtain models of according subjects. The models were evaluated by applying standard statistical classifier evaluation metrics. The results demonstrate the feasibility of the proposed system and classification approach.

Despite the low response-rate to the calls for participation of the data collection, enough data could be collected in order to perform an evaluation. Convincing people to participate in a data collection like this is a time-consuming task due to the private nature of the collected information. This might as well be a result of the reached audience, which is mainly comprised of computer science students from Germany. Germany has a long tradition in privacy related topics with respect to data collections, which might have made the endeavour more complicated. Unfortunately, the goal of collecting two weeks worth of context data could not be achieved for all participants. However, this circumstance did not have a measurable impact on the classification performance, which did not correlate with the training set size across models. The data collection process, as well as privacy aspects, challenges and the employed sensors were described in detail. Descriptive statistics of the participants demonstrate the variability of Android versions and devices that had to be supported by the data collection application. A final evaluation of the collected data revealed that very few instances of the noise feature were recorded and that the amount of measurements between features varies for each subject. Furthermore, the application of a metric, that expresses the estimated average degree of completeness, revealed, that up to 70% of the data of some subjects was lost during the collection process.

The theory of support vector machines, which is a supervised learning algorithm for binary classification tasks, was introduced to the reader in order to lay the foundation for the subject-specific modelling. The results obtained by training an SVM with the collected contextual information encourage further research in this area.

All models except two had precision scores of over 70%, the majority even a precision of over 80%. Five of the models had an F_1-score of over 70%, four of over 80%. The low F_1-score of the other models is a result of the low recall scores, i.e. too many false negatives. A low recall score indicates, that not enough discriminative features are included in the model, or that the existing features are correlated between the subjects. By gradually increasing the training set and evaluating the resulting model on a fixed test set, it was shown that a smaller training set can lead to a comparable performance and shorter training time of the classifier, which is especially important for the implementation of the proposed system on a mobile device.

In addition, the features that contribute most to the classification decision were retrieved from the computed models. A general feature selection based on the weights of all models is not feasible, since the weight distributions vary greatly among these. However, by building a sum over the feature weights of all models, it became obvious that behavioural characteristics have a higher total weight compared to the data obtained by physical sensors. A comparison between the average feature weights of good and bad performing models revealed, that the good performing models put a high weight on the location feature, while bad performing models put a high weight on the call duration feature. This result requires further investigations, to determine whether the good models adapt to the variance introduced by device specific sensor characteristics, or if they model a subject-specific behaviour.

Furthermore, the correlation between individual features was computed in order to reveal redundancy in the data set. An unsurprising result of this test is a strong correlation of the individual components of features extracted from physical sensor data. For instance, data for the individual axes of the accelerometer or magnetic field sensor are strongly correlated. Other examples for correlation are the latitude and longitude, and the three features associated with the calling behaviour of a subject. There was no significant correlation between seemingly unrelated features, which is a desirable result.

8.1. Future Work

Future work will be comprised of utilising more sophisticated features. For instance, instead of using the mean and median values of the accelerometer data, activity recognition could be performed to discriminate between different activities. This fits well with the result, that behavioural characteristics had a strong overall contribution in the evaluation of all models. By employing more sophisticated and additional behavioural features in the context model instead of simple statistical ones, the overall performance might improve. Other strategies for creating the training examples and replacing missing data could be tested, in order to evaluate the influence on the performance of the system. Instead of using the fixed values for the hyper parameters of the SVM, these should be optimised by performing a grid search. Since the evaluation in the presented thesis is performed on a desktop computer, an implementation and evaluation of the presented algorithms on a mobile device is recommended.

In addition to improvements to the context subsystem, an implementation of the missing components of the proposed architecture is desirable. This mainly includes the authentication subsystem, which is fully described by the BioAPI standard. By implementing the respective interfaces, the proposed architecture could be used as the foundation for a standard compliant new version of MBASSy. Furthermore, by implementing all of the proposed components, an SDK could be built, that can be used by developers as an authentication plugin for new applications. This resolves the issue of the current implementation of MBASSy, which is restricted to the demonstration of authentication algorithms.

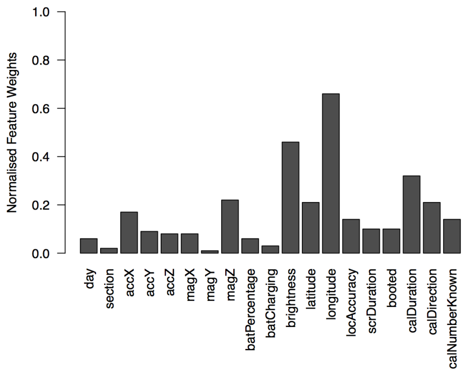
A. Glossary

API – Application Programming Interface
BFP – Biometric Function Provider
BSP – Biometric Service Provider
CASED – Center for Advanced Security Research Darmstadt
CPU – Central Processing Unit
DET – Detection Error Tradeoff
DSL – Domain Specific Language
EER – Equal Error Rate
FAR – False Accept Rate
FMR – False Match Rate
FNMR – False Non-Match Rate
FRR – False Reject Rate
FTA – Failure To Acquire
FTE – Failure To Enrol
GPS – Global Positioning System
HCI – Human Computer Interaction
HMM – Hidden Markov Model
IEC – International Electrotechnical Commission
ISO – International Organization for Standardization
JSON – JavaScript Object Notation
JTC – Joint Technical Committee
KKT – Karush-Kuhn-Tucker
MBASSy – Modular Biometric Authentication Service System
NFC – Near Field Communication
PC – Personal Computer
PIN – Personal Identification Number
QP – Quadratic Programming
RBF – Radial Basis Function
SDK – Software Development Kit

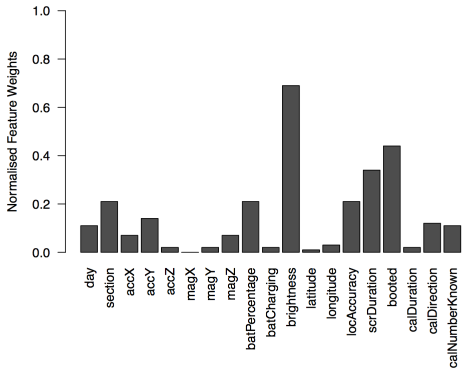
SIGCHI – Special Interest Group on Computer-Human Interaction
SIM – Subscriber Identity Module
SMS – Short Message Service
SSID – Service Set Identifier
SSL – Secure Sockets Layer
SVM – Support Vector Machine
ubicomp – Ubiquitous Computing
UDID – Unique Device Identifier
UIDAI – Unique Identification Authority of India
USB – Universal Serial Bus
UUID – Universally Unique Identifier
VPN – Virtual Private Network

B. Additional Evaluation Results

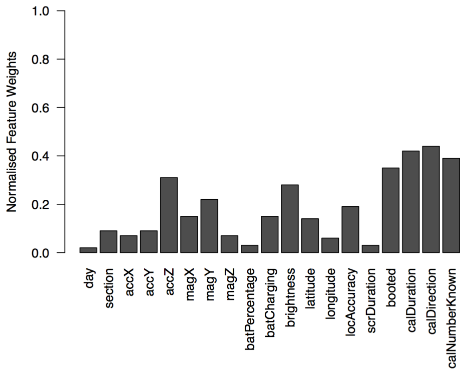
The following pages contain the normalised feature weight distributions for all generated models, as well as performance evaluation charts (Detected Curve, Lift Chart, ROC Curve) generated by the *KXEN InfiniteInsight* software.



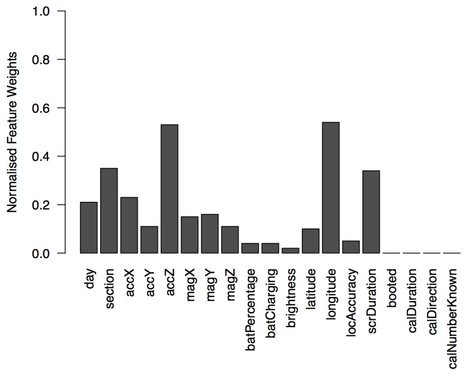
(a) Normalised Feature Weights Model 1 (F_1-score 0.46)



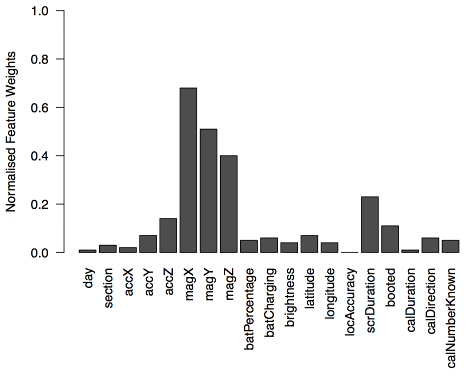
(b) Normalised Feature Weights Model 2 (F_1-score 0.56)



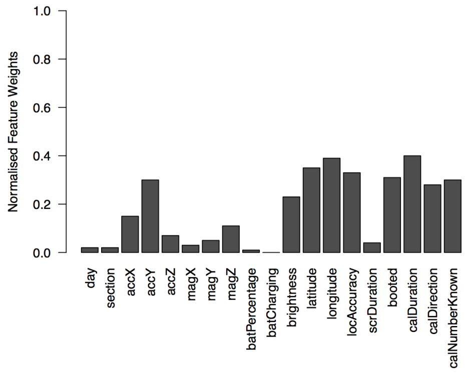
(c) Normalised Feature Weights Model 3 (F_1-score 0.25)



(d) Normalised Feature Weights Model 4 (F_1-score 0.56)

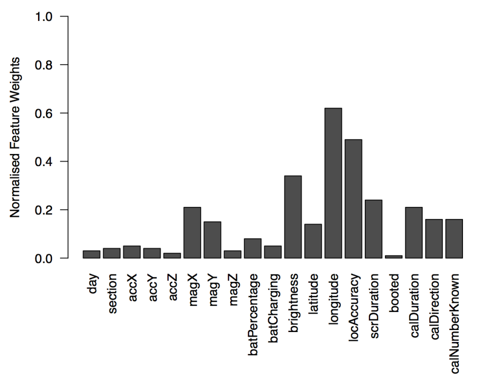


(e) Normalised Feature Weights Model 5 (F_1-score 0.88)

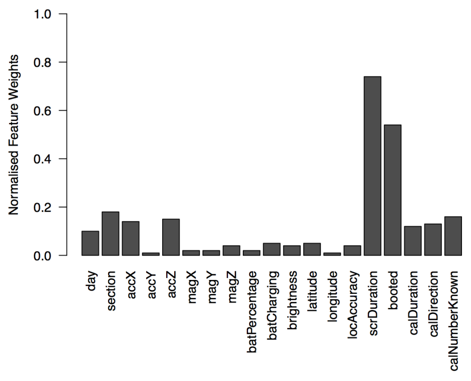


(f) Normalised Feature Weights Model 6 (F_1-score 0.9)

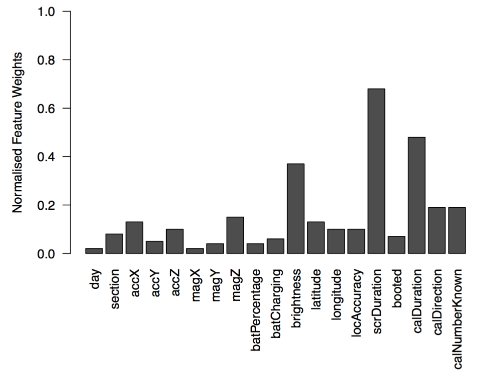
Figure B.1.: Normalised Feature Weights



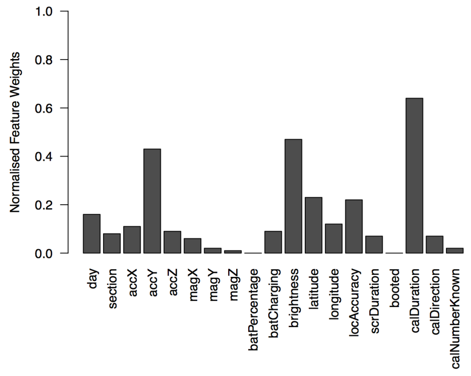
(a) Normalised Feature Weights Model 7 (F_1-score 0.9)



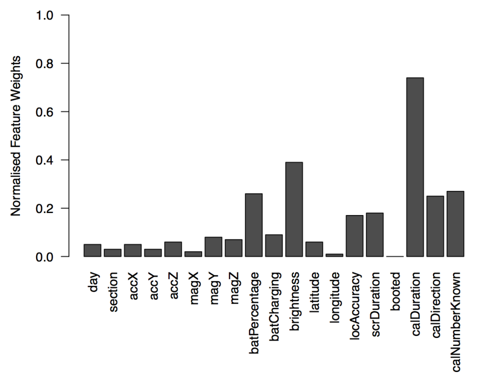
(b) Normalised Feature Weights Model 8 (F_1-score 0.33)



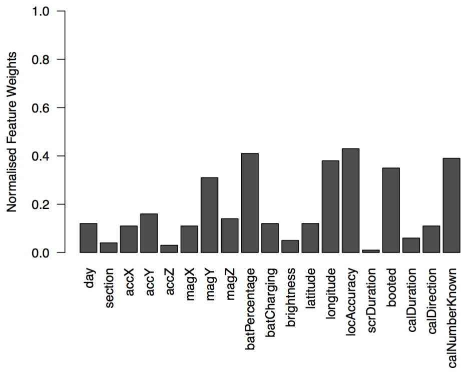
(c) Normalised Feature Weights Model 9 (F_1-score 0.47)



(d) Normalised Feature Weights Model 10 (F_1-score 0.3)

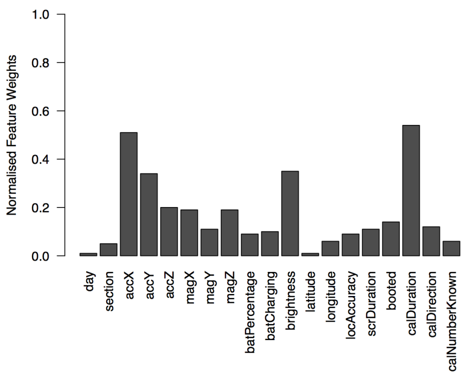


(e) Normalised Feature Weights Model 11 (F_1-score 0.3)

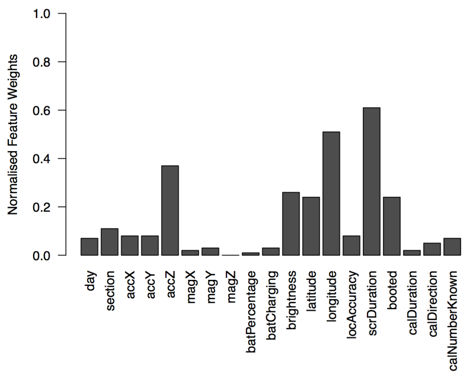


(f) Normalised Feature Weights Model 12 (F_1-score not available)

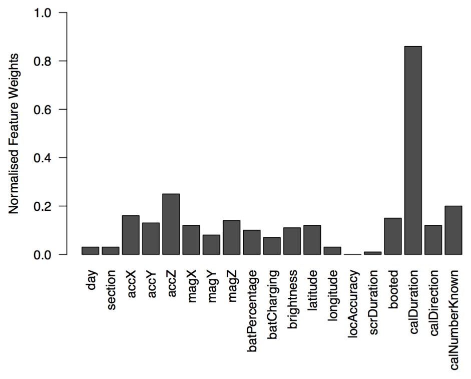
Figure B.2.: Normalised Feature Weights



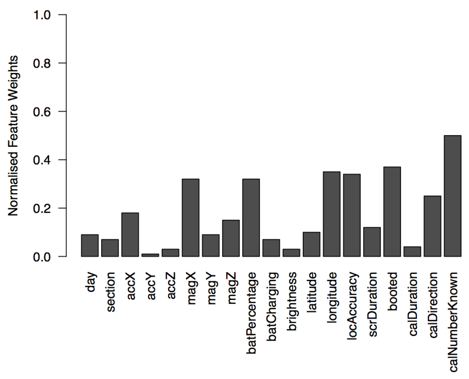
(a) Normalised Feature Weights Model 13 (F_1-score 0.56)



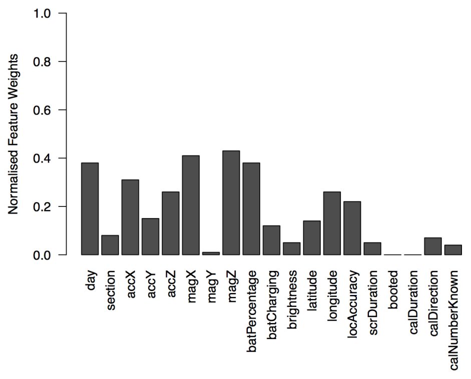
(b) Normalised Feature Weights Model 14 (F_1-score 0.72)



(c) Normalised Feature Weights Model 15 (F_1-score 0.55)

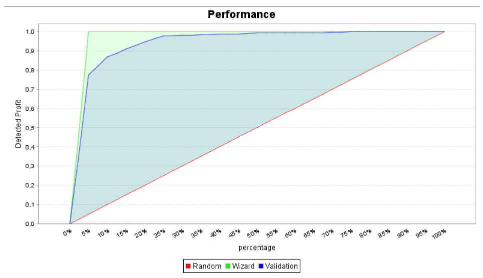


(d) Normalised Feature Weights Model 16 (F_1-score not available)

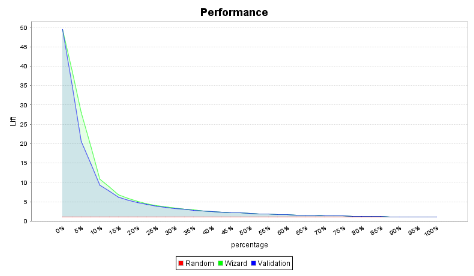


(e) Normalised Feature Weights Model 17 (F_1-score 0.83)

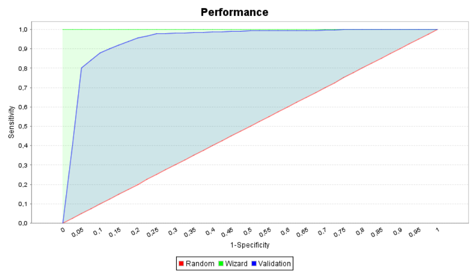
Figure B.3.: Normalised Feature Weights



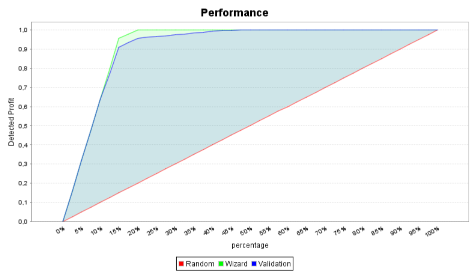
(a) Detected Curve for Model 3



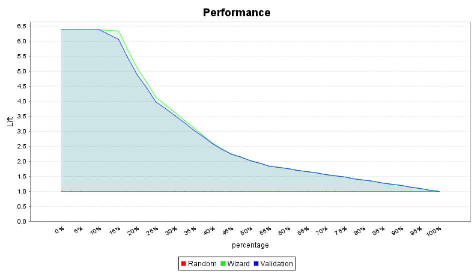
(b) Lift Chart for Model 3



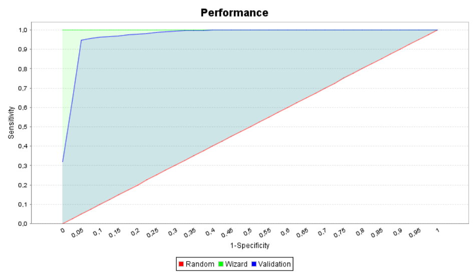
(c) ROC Curve for Model 3



(d) Detected Curve for Model 5

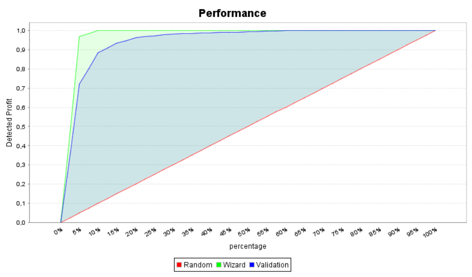


(e) Lift Chart for Model 5

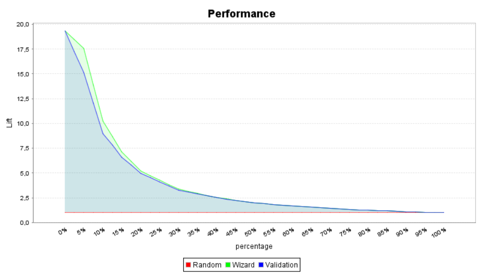


(f) ROC Curve for Model 5

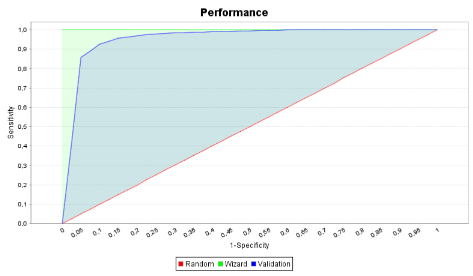
Figure B.4.: KXEN performance evaluation results



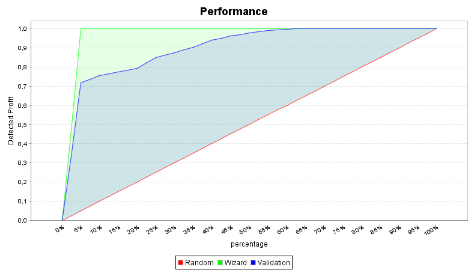
(a) Detected Curve for Model 6



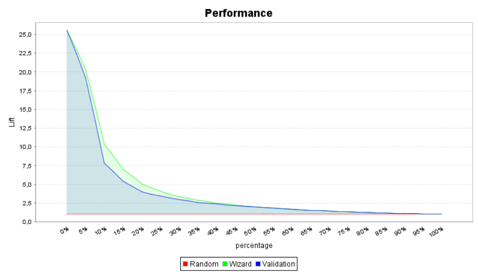
(b) Lift Chart for Model 6



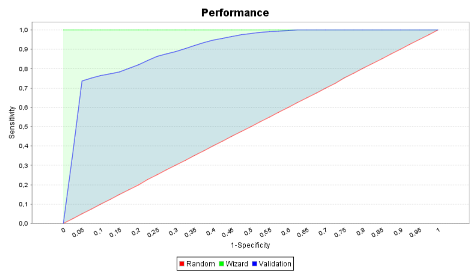
(c) ROC Curve for Model 6



(d) Detected Curve for Model 10

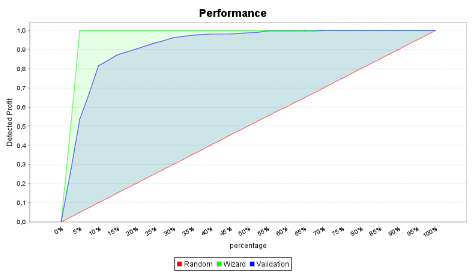


(e) Lift Chart for Model 10

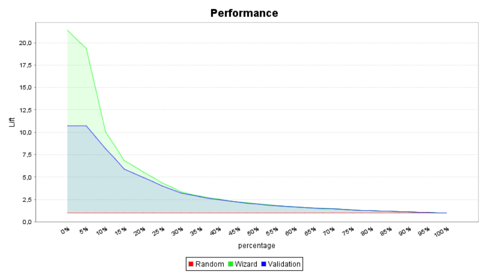


(f) ROC Curve for Model 10

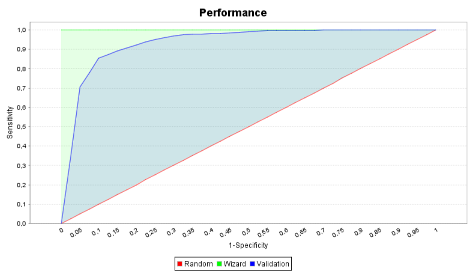
Figure B.5.: KXEN performance evaluation results



(a) Detected Curve for Model 12



(b) Lift Chart for Model 12



(c) ROC Curve for Model 12

Figure B.6.: KXEN performance evaluation results

C. Terms and Conditions of the Data Collection

C.1. Voluntary participation in the collection of contextual data

I participate voluntarily in the collection of contextual data. By using this software, I agree to the collection and use of my data as described below.

Data processing site which processes the collected data, is the Hochschule Darmstadt. The Hochschule Darmstadt ensures that the data will be used solely for the purposes of teaching and research. I have been informed that I can refuse my participation for any reason. I will use the software as described and participate at my own risk. I understand that I may inspect the collected data before they are used for teaching and research purposes. The insight is only possible by indicating the user ID, which is included in the software.

I was informed about the content and purpose of the data collected about me in written form. My data will be used only for the purposes described below.

All data collected will be deleted or reduced so that any relation to my person is irrevocably removed, as soon as they are no longer needed, but no later than after completion of the research.

C.2. Background information on the consent form

Description of the project purpose

After the installation and acceptance of these terms and conditions, context data will be collected in 20 minute intervals from each participant. The following data is collected continuously and sent over an encrypted connection to a server for storage:

- Anonymous subscriber information (gender, age group, occupation)

- Battery level and charging status
- Acceleration data
- Magnetic field strength
- Ambient brightness
- Ambient noise levels
- Encrypted Wi-Fi network name (SSID hash)
- Position data (based on network information, if available GPS)
- Time of the (de) activation of the screen
- Time from the start and shut down of the device
- Time of incoming and outgoing calls and information whether the number is known (the numbers themselves are not stored)

These data are used for the following purposes: Anaylsis of the data for model building and software development Publication of the records to the h_da biometric research group in the CASED (Center for Advanced Security Research)

The aim of the work in this thesis (MSc) is the improvement of mobile biometric systems by using contextual information. The data are used to generate a profile of the user, which allows the system to detect deviations from the normal usage. This will enable a more intelligent control of authentication processes. Security measures such as the PIN are used rarely, since they represent an impairment in the use of the devices. By using contextual information, an authentication will only be required, when an abnormal behavior is detected by the system.

Bibliography

[AD99] G Abowd and A Dey. Towards a better understanding of context and context-awareness. *Handheld and ubiquitous computing*, pages 304–307, 1999.

[App13] Apple Inc. Portrait-landscape rotation heuristics for a portable multifunction device, US-Pat. 7978176 [online]. Last Accessed: 03.02.2013. URL: `http://patft.uspto.gov/netacgi/nph-Parser?Sect2=PTO1&Sect2=HITOFF&p=1&u=/netahtml/PTO/search-bool.html&r=1&f=G&l=50&d=PALL&RefSrch=yes&Query=PN/7978176`.

[BC99] C J C Burges and D J Crisp. Uniqueness of the SVM Solution. In *Proceedings of the 1999 Conference on Advances in Neural Information Processing Systems (NIPS)*, pages 223–229, 1999.

[BN10] F Breitinger and C Nickel. User Survey on Phone Security and Usage. In *BIOSIG 2010 - Proceedings of the Special Interest Group on Biometrics and Electronic Signatures, 09.-10. September 2010 in Darmstadt, Germany*, pages 139–144, 2010.

[Bur98] C J C Burges. A Tutorial on Support Vector Machines for Pattern Recognition - Springer. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

[BV09] S Boyd and L Vandenberghe. *Convex optimization.* Number 978-0521833783. Cambridge University Press, 2009 edition, 2009.

[Cov65] T M Cover. Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition. *IEEE Transactions on Electronic Computers*, EC-14(3):326–334, 1965.

[DWMB12] M Derawi, H Witte, S McCallum, and P Bours. Biometric access control using Near Field Communication and smart phones. *5th IAPR International Conference on Biometrics (ICB)*, pages 490–497, 2012.

[FLBG10] M Ferras, C-C Leung, C Barras, and J L Gauvain. Comparison of Speaker Adaptation Methods as Feature Extraction for SVM-Based Speaker Recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(6):1366–1378, 2010.

[Goo13] Google. Android Developer Reference [online]. Last Accessed: 03.02.2013. URL: `http://developer.android.com/reference/packages.html`.

[HBC$^+$13] T T Hewett, R Baecker, S Card, T Carey, J Gasen, M Mantei, G Perlman, G Strong, and W Verplank. ACM Special Interest Group on Computer-Human Interaction (SIGCHI) Curricula for Human-Computer Interaction [online]. Last Accessed: 03.02.2013. URL: `http://old.sigchi.org/cdg/cdg2.html#2_1`.

[HBL$^+$07] R J Hulsebosch, M S Bargh, G Lenzini, P W G Ebben, and S M Iacob. Context sensitive adaptive authentication. In *Proceedings of the 2nd European conference on Smart sensing and context*, EuroSSC'07, pages 93–109. Springer-Verlag, October 2007.

[ISO06a] ISO. ISO/IEC 19784-1:2006 Information technology – Biometric application programming interface – Part 1: BioAPI specification, 2006.

[ISO06b] ISO. ISO/IEC 19795-1:2006 Information Technology — Biometric Performance Testing and Reporting — Part 1: Principles and Framework , 2006.

[ISO10] ISO. ISO/IEC JTC1/SC 37 Standing Document 11 (SD 11), Part 1 Harmonization Document, 2010.

[ISO11] ISO. ISO/IEC 24745:2011 Information technology – Security techniques – Biometric information protection, 2011.

[ISO12a] ISO. ISO/IEC 2382-37:2012 Information Technology – Vocabulary – Part 37 : Biometrics, 2012.

[ISO12b] ISO. ISO/IEC CD 30106-1 BioAPI for Object Oriented Programming Languages, 2012.

[JGVH95] R Johnson, E Gamma, J Vlissides, and R Helm. *Design Patterns: Elements of Reusable Object-Oriented Software.* Number 978-0201633610. Addison-Wesley, 1995.

[Lam78] L Lamport. The implementation of reliable distributed multiprocess systems. *Computer Networks (1976)*, 2(2):95–114, 1978.

[LRAD11] J C D Lima, C C Rocha, I Augustin, and M A R Dantas. A Context-Aware Recommendation System to Behavioral Based Authentication in Mobile and Pervasive Environments. In *IFIP 9th International Conference on Embedded and Ubiquitous Computing (EUC)*, pages 312–319, 2011.

[LZC$^+$11] Y Li, B Zhang, Y Cao, S Zhao, Y Gao, and J Liu. Study on the Bei-Hang Keystroke Dynamics Database. In *International Joint Conference on Biometrics IJCB*, pages 1–5, 2011.

[Mer09] J Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, pages 415–446, 1909.

[Mos13] Marie-Luise Moschgath. *Kontextabhängige Zugriffskontrolle für Anwendungen im Ubiquitous Computing.* PhD thesis, TU Darmstadt, Darmstadt, July 2002, Last Accessed: 03.02.2013. URL: `http://tuprints.ulb.tu-darmstadt.de/epda/000333/Moschgath-Diss.pdf`.

[NBB11] C Nickel, H Brandt, and C Busch. Classification of acceleration data for biometric gait recognition on mobile devices. *BIOSIG 2011-Proceedings of the Special Interest Group on Biometrics and Electronic Signatures*, pages 57–66, 2011.

[NBRM11] C Nickel, C Busch, S Rangarajan, and M Möbius. Using Hidden Markov Models for Accelerometer-Based Biometric Gait Recognition. *7th International Colloquium on Signal Processing and its Applications (CSPA)*, pages 58–63, 2011.

[Ng13] Andrew Ng. CS229 Lecture notes - Support Vector Machines [online]. Last Accessed: 03.02.2013. URL: `http://cs229.stanford.edu/notes/cs229-notes3.pdf`.

[NWB12] C Nickel, T Wirtl, and C Busch. Authentication of Smartphone Users Based on the Way They Walk Using k-NN Algorithm. In *Eighth International Con-*

ference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), pages 16–20. IEEE, 2012.

[Pas98] J Pascoe. Adding generic contextual capabilities to wearable computers. In *Proceedings of the 2nd IEEE International Symposium on Wearable Computers*, ISWC '98, pages 92–99, 1998.

[RJ86] L Rabiner and B Juang. An introduction to Hidden Markov Models. *IEEE ASSP Magazine*, 3(1):4–16, 1986.

[SAW94] B Schilit, N Adams, and R Want. Context-Aware Computing Applications. *First Workshop on Mobile Computing Systems and Applications (WMCSA)*, pages 85–90, 1994.

[SBG99] A Schmidt, M Beigl, and H W Gellersen. There is more to context than location. *Computers & Graphics*, 23(6):893–901, 1999.

[SNJC11] E Shi, Y Niu, M Jakobsson, and R Chow. Implicit authentication through learning user behavior. In *Proceedings of the 13th international conference on Information security*, ISC'10, pages 99–113, 2011.

[SS98] P Samarati and L Sweeney. Generalizing data to provide anonymity when disclosing information (abstract). In *Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, PODS '98, pages 188–202, May 1998.

[SS11] C Soanes and A Stevenson. *Concise Oxford English Dictionary.* Number 978-0199601-080. Oxford University Press, 12 edition, 2011.

[Vap82] V Vapnik. *Estimation of Dependences Based on Empirical Data: Springer Series in Statistics (Springer Series in Statistics).* Number 978-0-387-30865-4. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1 edition, 1982.

[Vap99] V N Vapnik. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, 10(5):988–999, 1999.

[Wei91] M Weiser. The computer for the 21st century. *Scientific American*, pages 94–104, 1991.

[Wei93a] M Weiser. Hot topics-ubiquitous computing. *Computer*, 26(10):71–72, 1993.

[Wei93b] M Weiser. Some computer science issues in ubiquitous computing. *Communications of the ACM*, 36(7):75–84, 1993.

[WLL+09] J Wang, Y Li, P Liang, G Zhang, and A Xinyu. An effective multi-biometrics solution for embedded device. In *IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pages 917–922, 2009.

[WN10] H Witte and C Nickel. Modular Biometric Authentication Service System (MBASSy). In *BIOSIG 2010 - Proceedings of the Special Interest Group on Biometrics and Electronic Signatures, 09.-10. September 2010 in Darmstadt, Germany*, pages 115–120, 2010.

[WN11] T Wirtl and C Nickel. Aktivitätserkennung auf Smartphones. In *BIOSIG 2011 - Proceedings of the Special Interest Group on Biometrics and Electronic Signatures, 08.-09. September 2011 in Darmstadt, Germany*, pages 195–202, 2011.

[XS02] N Xiong and P Svensson. Multi-sensor management for information fusion: issues and approaches. *Information Fusion*, 3(2):163–186, June 2002.

[YFEE10] Z Yaghoubi, K Faez, M Eliasi, and A Eliasi. Multimodal biometric recognition inspired by visual cortex and Support vector machine classifier. In *International Conference on Multimedia Computing and Information Technology (MCIT)*, pages 93–96, 2010.