FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Design and Implementation of an Automatic Tourist Guide

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieurin

im Rahmen des Studiums

Software Engineering and Internet Computing

eingereicht von

Manuela Weilharter

Matrikelnummer 0826002

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuer: Ao. Univ.Prof. Mag. Dr. Horst Eidenberger

Wien, 03.03.2015

(Unterschrift Verfasserin) (Unterschrift Betreuer)

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Design and Implementation of an Automatic Tourist Guide

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieurin

in

Software Engineering and Internet Computing

by

Manuela Weilharter

Registration Number 0826002

to the
Faculty of Informatics at the Vienna University of Technology

Advisor: Ao. Univ.Prof. Mag. Dr. Horst Eidenberger

Vienna, 03.03.2015 _____ _____

(Signature of Author) (Signature of Advisor)

Technische Universität Wien

A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.ac.at

Erklärung zur Verfassung der Arbeit

Manuela Weilharter
Rosasgasse 13, 1120 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum) (Unterschrift Verfasserin)

Acknowledgements

First of all I want to thank my supervisor Horst Eidenberger for the opportunity to work on such an interesting topic and for his continuous support and guidance throughout the process of creating this master thesis.

A special thanks goes to my friends Markus Lanner and Stephan Teuschl who repeatedly tested my application and gave me a lot of important feedback and suggestions on how to improve it.

I am also grateful to Florian Kimmel for his encouragement and helpful advice when I got stuck and to my friends and family who took my mind of things.

Abstract

Over the last years, smartphones have evolved to an important computing platform. Cities therefore offer mobile applications to provide visitors with guidance and support to boost the tourism industry. There are a lot of available tourist applications for Vienna, but they tend to focus on accommodation, navigation, social services or marketing instead of providing information for sightseeing.

This master thesis focuses on the sightseeing aspect for tourists and presents an application for the Android platform which offers location-based information on sights in Austria, Germany, Switzerland and Liechtenstein. A back-end information retrieval (IR) application was created to gather the content automatically from Wikipedia and to provide an interface for the front-end to obtain the data with an update. This enables an offline usage of the Android application to avoid roaming charges.

The main feature of the Automatic Tourist Guide (ATG) is the intuitive guide which makes it possible for the user to walk through the city enjoying the view while the application delivers the relevant information via text-to-speech whenever a point of interest (POI) is passed. This is achieved by determining the exact location of the user and then computing the nearby POIs using a distance calculation algorithm. Furthermore, the ATG provides a map with all available sights and the route to the nearest attraction. The user is also able to choose categories in which he is interested in and will be presented only with content regarding his preferred subjects. In addition, there are three different versions of the content available: the unabridged text, a summarization of the most valuable information and a brief overview with only one paragraph.

Kurzfassung

Smartphones haben in den letzten Jahren einen großen Aufschwung erlebt. Sie werden nicht nur zum Telefonieren verwendet, die Benutzer surfen damit im Internet, rufen E-Mails ab und geben Bestellungen auf. Für viele Städte sind deshalb mobile Anwendungen verfügbar, die verschiedenste Services für Touristen anbieten. Für Wien gibt es bereits eine Menge an verfügbaren Tourismus-Anwendungen, welche sich jedoch hauptsächlich auf das Finden einer Unterkunft, Navigation, soziale Dienste und Marketing konzentrieren. Anwendungen, die sich auf Sightseeing spezialisieren, gibt es nur wenige.

Diese Masterarbeit konzentriert sich auf den Sightseeing-Aspekt für Touristen und präsentiert eine Anwendung für die Android-Plattform, die ortsbezogene Informationen über Sehenswürdigkeiten (sogenannten points of interest - POIs) in Österreich, Deutschland, Schweiz und Liechtenstein anbietet. Es wurde eine Webanwendung im Bereich Information Retrieval (IR) entwickelt, die den Inhalt automatisch von Wikipedia extrahiert und über ein Webservice dem Front-End als einmaliges Update zur Verfügung stellt. Dies ermöglicht eine Benutzung der Android-Anwendung auch ohne Internetverbindung.

Das zentrale Feature des Automatic Tourist Guide (ATG) ist der automatische Stadtführer. Dieser ermöglicht dem Benutzer durch die Stadt zu spazieren, während die Anwendung, sobald eine Sehenswürdigkeit in die Nähe kommt, die Informationen darüber automatisch vorliest. Dazu wird die exakte Position des Benutzers benötigt, wodurch alle Sehenswürdigkeiten in der Nähe ermittelt werden können. Dies geschieht mit Hilfe der Vincenty-Formel zur Berechnung der Distanz zwischen zwei Positionen. Der ATG bietet zusätzlich eine Übersichtskarte mit allen umliegenden POIs und eine Streckenbeschreibung zur nächstgelegenen Sehenswürdigkeit. Der Benutzer kann verschiedene Themengebiete wählen und wird dann nur mit Informationen über diese Themen versorgt. Zusätzlich gibt es verschiedene Versionen des Inhalts: eine ungekürzte Fassung, eine Zusammenfassung mit den nützlichsten Informationen und eine kurz gehaltene Übersicht.

Terms and Abbreviations

A-GPS Assisted Global Positioning System 15, 19

AoA Angle of Arrival 15, 17, 18

API Application Programming Interface 40, 49, 50, 65, 67, 68, 72, 73

AR Augmented Reality 4

ATG Automatic Tourist Guide iii, iv, 1, 3–6, 11, 20, 21, 26, 30, 34, 36, 38–42, 44, 47, 49, 50, 52, 55, 57–59, 63, 65, 66, 68, 73–77

ATS Automatic Text Summarization 26

BS Base Station 16–18

BTS Base Transceiver Station 16

Cell-ID Cell Identification 14, 16

CRUD Create, Read, Update, Delete 60

DMS Degree/Minute/Seconds 8

ECEF Earth Centered Earth Fixed 7

GPS Global Positioning System 14–16, 18, 19, 73

GSM Global System for Mobile Communications 18

GUI Graphical User Interface 43, 50, 53, 56, 63, 65, 72, 76, 77

HTML Hypertext Markup Language 65, 66, 68

HTTP Hypertext Transfer Protocol 58, 63, 66, 67, 70

IDE Integrated Development Environment 62

IE Information Extraction 3

IR Information Retrieval iii, 2, 21, 22, 40, 75

JSF JavaServer Faces 60, 65, 76

JSON JavaScript Object Notation 65, 67

LAI Local Area Identity 16, 17

LBS Location-based Services 2–6, 8, 73

LLR Log-likelihood Ratio 28

MVC Model–View–Controller 58, 60

NLP Natural Language Processing 2, 4, 5, 21, 24, 65, 75

POI Point of Interest iii, iv, 1–6, 39–50, 52–56, 60, 63–66, 69–73, 76, 77

POS Part-of-Speech 28, 35, 65

RSSI Received Signal Strength Intensity 16

SOAP Simple Object Access Protocol 76

SSID Service Set Identifier 16

SVM Support Vector Machines ix, 30, 32

TA Timing Advance 15, 18

TDoA Time Difference of Arrival 14, 17

TF-IDF Term Frequency Inverse Document Frequency 23, 28

ToA Time of Arrival 14, 17

TTFF Time to First Fix 16, 19

TTS Text-to-Speech 2, 34–38, 44–46, 50, 55, 63, 72

UML Unified Modeling Language 58, 60

URL Uniform Resource Locator 70, 77

WLAN Wireless Local Area Network 16, 73

Contents

CHAPTER 1

Introduction

The Automatic Tourist Guide is an Android application designed for people who want to explore a city by learning about the buildings and structures in it. It offers points of interest (POIs) and further information about them in a visual as well as an acoustic way. Additionally, it places the user on a map and displays the surrounding tourism attractions. This chapter explains the motivation, goals and contribution of the ATG and presents related works. It concludes by providing an overview over the structure of this thesis.

1.1 Motivation and problem statement

Smartphones have evolved to an important computing platform in the last years. One key element behind their popularity is the versatility. People use them not only to make calls and send texts, they browse the Web, play games or even perform banking transactions. Communities therefore offer smartphone applications to provide visitors with guidance and support to boost the tourism industry. There are a lot of available tourist applications for Vienna. On one hand, there are those which concentrate on visual output. They focus on providing information to hotels, bars and restaurants and offer user reviews and booking options rather than providing information for sightseeing. On the other hand, there are applications which provide audio information about sights like a pocket guide. They offer tours which you can buy. These tours consist of predefined routes with accompanying audio information and the user can listen to the prerecorded text while he is guided through the city.

This master thesis presents an application for the Android platform which offers location-based information about the city by extracting relevant knowledge from the web. The user is able to walk through the city enjoying the view, while the application delivers the relevant information whenever a POI is passed.

The content is gathered on an initial update and stored in a database, so that it is available offline and the usage of the application causes no data roaming charges. Each section of the text is classified and presented for the users not only in a visual, but furthermore in an acoustic way via text-to-speech (TTS). As another feature, information retrieval methods have been used to create different versions of the text to ensure that it appeals to a variety of users. There is the normal text which is basically the cleaned up version of the Wikipedia page. As a second level, a summarization of the text is available which presents the most significant sentences of the content. At last, the user can choose the short text level where the POI is presented in one compact paragraph.

Additionally, the user is able to switch between different forms of guides: architecture, history, sports and geography. As an example, if the user has chosen the category history and walks by Schönbrunn Palace, she will receive information about when it was built and what it was used for during the centuries. However, if the chosen category is sports, other information is more suitable. For example Schönbrunn also includes a beautiful garden where runners are allowed.

1.2 Related work

This master thesis addresses a couple of different problems which have been discussed in scientific literature before. The three basic topics are location-based services, natural language processing and the usage of mobile devices as tourist guides. This section provides an overview of the existing findings and outlines how this thesis differs from them.

1.2.1 Location-based services

Location-based services (LBS) have been frequently discussed over the years. Research into the different methods for mobile positioning is a focal point in several articles. Adusei et al. present an overview of the most common positioning techniques and analyze them with regard to performance metrics like accuracy, reliability and applicability [2]. Zeimpekis et al. discuss the same topic in their paper with a focus on the accuracy of the different techniques [68].

Besides accuracy, another challenge for LBS is their power consumption. In [45] Wolfgang Narzt presents an energy saving model for mobile location-based services. Eberle and Perrucci

analyze the power consumption of several positioning strategies and discuss the trade-off between precision and energy efficiency in [14].

This master thesis aims to provide the necessary theoretical background knowledge on location-based services including the different techniques for positioning and their benefits and drawbacks as well as methods for calculating the distance between two positions. These topics are basic principles for the ATG as it requires an accurate positioning of the user and a precise distance calculation to the surrounding POIs.

1.2.2 Natural language processing

Extracting relevant data from a web page has been the topic of numerous scientific papers and books. Several knowledge bases (KBs) using Wikipedia as the main source are available to fit this need, for example the semantic knowledge base YAGO. It extracts information from Wikipedia using category pages which combine articles belonging to a specific category [59]. For example Schönbrunn Palace can be found in the category *baroque palaces*. Another information extraction approach is proposed by Wu and Weld. They present an open information extraction (IE) system that uses a new form of self-supervised learning, based on a heuristic match between Wikipedia infoboxes and corresponding text [66].

Miltsakaki and Troutt present a tool called Read-X that performs a real-time web text extraction, detects the reading difficulty and classifies the text into categories like art, literature, philosophy, sports, religion and others. For the content classification they use a mix of different classifiers including a Naive Bayes and a Maximum Entropy classifier [43].

The technique for the ATG is a combination of the above-mentioned approaches. First, the program extracts all page IDs for a specific country by traversing Wikipedia's category system recursively. In a second step, the content for each page is extracted, cleaned up and classified using a Maximum Entropy classifier.

1.2.3 Tourist Guides

Using mobile phones as electronic tourist guides is a not a new concept. POInter was an early tourist guide system developed by Hill and Wesson for Microsoft Windows 5.0 based Pocket PC Phone Edition devices. It concentrates on finding POIs according to the users preferences [23]. More recently Alves et al. developed KUSCO (Knowledge Unsupervised Search for instantiating Concepts on lightweight Ontologies), a system which searches the web for related pages to a given POI using semantic enrichment of the data [6].

Michele et al. present in [42] a tourism application called VisitAR which uses augmented reality (AR) to enrich the appearance. The focus of this software lies in the visual presentation of the information. They included photos and videos of the POIs and offer a 3D map of the city.

The Automated Tourist Guide deviates from the existing guides in several aspects. First, the ATG focuses on the automated information extraction from Wikipedia. It does not require the content to be entered manually into a database; instead the back-end application automatically mines Wikipedia and makes the cleaned up text available for the ATG via an update. Another special characteristic is the audio component which reads the information about the POI to the user. Furthermore different versions of the content are available to suit the users needs.

1.3 Methodological approach

The aim of this master thesis was to build a prototype application for a mobile platform which serves as a tourist guide in Vienna. The content should be obtained automatically from the web, cleaned up, classified, summarized and stored in a database.

These requirements led to a breakdown of the work into two separate applications: a back-end web application which automatically gathers the content from the web using Wikipedia as the main source and the front-end Android application. The back-end offers all data to the front-end over a RESTful web service.

The first step was to get an overview of already existing tourism applications for Vienna to get a sense of how the ATG differs from them. Testing these applications helped to identify popular features and lead to ideas for the implementation. The results of this analysis served as a basis for defining the use cases and the basic requirements for the ATG which were then formalized and refined.

Prior to the implementation, research into location-based services and natural language processing was required because the ATG depends on subareas of these fields. For example, an exact positioning of the user is important and so are the classification and the summarization of the Wikipedia text.

Finally, the ATG was implemented following current design guidelines for the Android platform. Afterwards, it was tested and the feedback and suggestions of the test users were incorporated into the final version of the application.

1.4 Structure of the work

Chapter 2 offers background information on location-based services which are used in the ATG to place the user on the map and to offer the appropriate POIs in his vicinity. Different methods for positioning and distance calculation are presented.

Chapter 3 provides theoretical information on the natural language processing (NLP) methods used to create structured and informative content for the application. After a general overview about information retrieval, the three main topics are text classification, text summarization and speech processing.

Chapter 4 discusses the design of the ATG which includes details on the requirements, the use cases and the final user interface. Furthermore, it presents the challenges which occurred within the scope of this master thesis.

Chapter 5 reveals details on the concrete implementation. It provides specifics about the architecture of two parts of the ATG: the back-end and the front-end.

Finally, Chapter 6 concludes the thesis and provides an outlook for future work based on the findings in the previous chapters.

CHAPTER 2

Location-based Services

Location-based services (LBS) are an approach to combine geographic location data with the general concept of services. They discover the position of the user in order to provide him with services fitting his location. Areas of application include emergency services, navigation, logistic monitoring and informational services such as news, weather or sports. The Automatic Tourist Guide uses location-based services to position the user on the map and to calculate nearby points of interest.

This chapter serves as theoretical background information for the design and implementation chapters of the ATG and starts off by describing the different methods for determining a location. Afterwards the most common approaches for calculating the distance between two positions are presented. The chapter concludes by providing an overview and a comparison of the different positioning techniques.

2.1 Location

When dealing with location-based services it is important to specify the term location. A location describes a specific position or point on the surface of the earth. Locations are based on geographic coordinate systems that divide the earth into units of the same size and shape. A datum represents the coordinate system and is utilized to establish the location. There are horizontal and vertical datums. The former defines how coordinate system grids align with the earth's surface. The latter measures the height of a position with regard to the mean sea level. In

geodesy there are two common classes of coordinate systems: the Cartesian and the Ellipsoidal coordinate system [36].

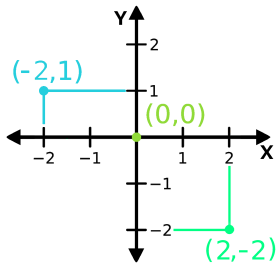

Figure 2.1: Axes of a two-dimensional Cartesian coordinate system.

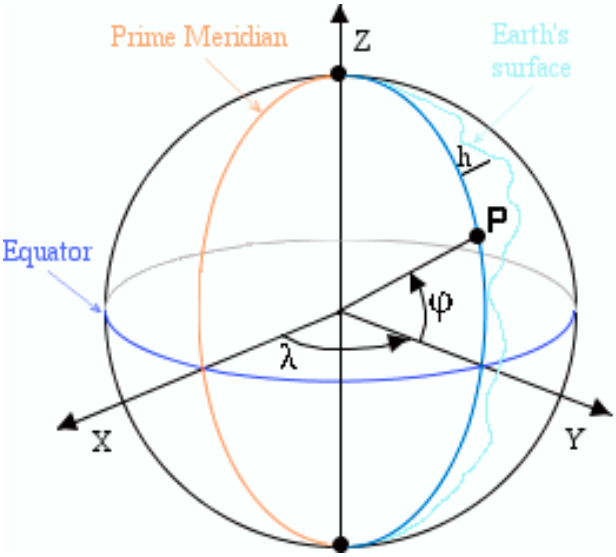

Figure 2.2: Coordinate system with latitude φ, longitude λ and height h and the corresponding Cartesian axes (X, Y, Z).

2.1.1 Cartesian coordinate system

In geodesy, the Cartesian coordinate system specifies each point uniquely by specifying its distance to the axes either in a two-dimensional (only X- and Y-axis) or in a three-dimensional space (including the Z-axis). The point where the axes meet is called origin and is specified as (0, 0) and (0, 0, 0), respectively. A pair of axes defines a plane which is labeled according to the corresponding axes. Figure 2.1 shows a simplified two-dimensional system with the two locations (-2, 1) and (2, -2) on the XY-plane which is called the equatorial plane [36].

The mostly used Cartesian system is the three-dimensional Earth Centered, Earth Fixed (ECEF) model which places the origin at the predicted center of the earth as shown in Figure 2.2. In the ECEF the X-axis points to the intersection of the prime meridian and the equator, while the Z-axis is aligned with the rotation axis of the earth (polar axis). The Y-axis completes the right-handed orthogonal system by forming right angles with the other two axes. Due to this constellation the ECEF rotates with the earth.

2.1.2 Ellipsoidal coordinate system

The Ellipsoidal coordinate system is a more common way of stating a position. Ellipsoid refers to the model approximating the shape of the earth and has its origin at the predicted earth's center. The Ellipsoidal coordinate system uses two angles, latitude and longitude, to specify the location. Figure 2.2 displays a location P with its latitude φ and longitude λ. To fully specify a location, the ellipsoidal height h is required which is the difference between the topographic exterior of the earth and the ellipsoid [54].

Latitude represents the north-south position of a location and is specified by the angle between the equatorial XY-plane and the straight line that goes from the origin to the location P. It is a convention that latitudes north of the equator are positive and south ones are negative. Lines of latitude go from 0° to 90° north and 0° to -90° south, the equator is located at 0° and the North/South Poles can be found at 90°/-90°.

Longitude refers to the east-west position and is specified by the angle east or west from the prime meridian. The longitude measurements range from 0° to 180° to the east and 0° to -180° to the west from the prime meridian [70].

Latitude and longitude can be expressed in two different ways: the decimal format and the degree/minute/seconds (DMS) format. The decimal representation can include eight decimal places and pinpoints a location to within one millimeter. The DMS format uses degrees, minutes and seconds to specify the position and requires furthermore a declaration of the direction (south, west, east or north) since there are no negative values in this format. As an example the Stephansdom is located at (48.2083, 16.3728) and accordingly (48° 12' 29.8800" N, 16° 22' 22.0800" E) in Vienna and the Christ the Redeemer statue is found at (-22.95158, -43.210482) and accordingly (22° 57' 5.6880" S, 43° 12' 37.7352" W) in Rio de Janeiro.

2.2 Distance

Calculating the distance between one locations and another is a common application of location-based services. Distance calculations can be used to determine the shortest path in navigation systems or to find the nearest points of interest in route planners. There are different ways for computing the distance depending on the coordinate system and datum. The most commonly used methods are [52]:

- Euclidean distance
- Manhattan distance

- Haversine formula
- Spherical Law of Cosines
- Vincenty's formula

2.2.1 Euclidean distance

The Euclidean or straight-line distance within two points is the length of a straight line connecting them. For two points with Cartesian coordinates (X_1, Y_1, Z_1) and (X_2, Y_2, Z_2) the distance d can be calculated by the formula:

$$d = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2 + (Z_2 - Z_1)^2}. \tag{2.1}$$

In geodesy, this formula should only be used when the two locations are near each other because it does not incorporate the asperity of the earth and is therefore imprecise for larger distances [52].

2.2.2 Manhattan Distance

The Manhattan distance, also known as taxicab geometry or city block distance, alludes to the grid layout of Manhattan's streets. It is the sum of the horizontal and vertical paths that connect the two points, similar to a car driving along the street grid. To be able to use the Manhattan formula in practice some simplifications have to be made. For example, the streets are assumed to run only in straight lines and have no width and are accessible in both ways. Furthermore it is supposed that there are no obstacles along the path and buildings are of point size. With these simplifications the calculation becomes relatively simple as the result is the sum of the absolute difference of the single coordinates [34]. For the two points (X_1, Y_1) and (X_2, Y_2) the following equation yields the distance:

$$d = |X_1 - Y_1| + |X_2 - Y_2|. \tag{2.2}$$

Figure 2.3 shows the difference between the Euclidean and the Manhattan distance. The two points are located at the coordinates (1, 1) and (5, 5). The red, the green and the yellow lines denote three different paths using the Manhattan distance and connect the two points with a length of 8. The straight blue line shows the Euclidean distance and has a length of $\sqrt{(5-1)^2 + (5-1)^2} = \sqrt{32} \approx 5.66$.

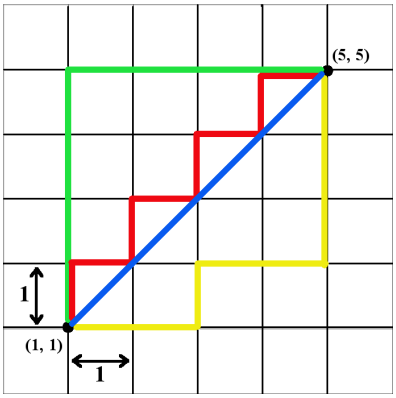

Figure 2.3: Euclidean (straight line) and three Manhattan distances.

Areas of application for the Manhattan distance calculation in spatial planning are all situations where a straight-line result is not desired. For example in route planning the Euclidean distance is not realistic since a person is constrained to travel along streets or sidewalks.

2.2.3 Haversine Formula

The Haversine formula computes the great-circle distance between two latitude and longitude coordinates. In contrast to straight line methods like Euclidean which cuts through the earth's interior, the great-circle distance is measured alongside the surface of the sphere. The Haversine formula was published by Sinnott in 1984 [57]. Unlike the Manhattan algorithm, it calculates the distance "in a beeline" which means ignoring any terrain information. The formula assumes a spherical earth which leads to an accuracy that can be out by 0.3% [52]. Given the coordinates of two points (X_1, Y_1) and (X_2, Y_2) the distance d is computed as follows:

$$d = 2 * r * \arcsin(\sqrt{\sin^2(\frac{X_2 - X_1}{2}) + \cos X_1 * \cos X_2 * \sin^2(\frac{Y_2 - Y_1}{2})}. \tag{2.3}$$

2.2.4 Spherical Law of Cosines

The spherical law of cosines is another distance calculation method which uses a spherical model of the earth. It is mathematically equivalent to Haversine, but when Sinnott published the formula in 1984 the computational power was limited. This fact lead to rounding errors with the law of cosines formula for small distances [29]. Nowadays computers process 64bit floating-point

numbers which provide a precision of up to 15 decimal digits. Therefore the law of cosines computes results with the same accuracy as the Haversine calculation [1]. The distance *d* computed using the law of Cosines is

$$d = r * \arccos(\sin Y_1 * \sin Y_2 + \cos Y_1 * \cos Y_2 * \cos(X_2 - X_1)) \tag{2.4}$$

where *r* is the earth's radius of 6371 kilometers [22].

2.2.5 Vincenty's Formula

Vincenty's formula is a very precise distance calculation because it considers that the earth's shape is an ellipsoid rather than a perfect sphere. This leads to very accurate results to within 0.5 millimeters. Vincenty first presented his algorithm in 1975 with regards to efficient programming through conserving space and reducing the execution time [62]. For the Automatic Tourist Guide this formula is used because it is important to calculate the distance to the surrounding points of interest as accurate and efficient as possible. For information about the concrete implementation of Vincenty's formula in the ATG please refer to Section 5.3.2. The distance using Vincenty's formula can be obtained by following equation:

$$d = r * \arctan\left(\frac{\sqrt{(\cos X_2 * \sin \Delta Y)^2 + (\cos X_1 * \sin X_2 - \sin X_1 * \cos X_2 * \cos \Delta Y)^2}}{\sin X_1 * \sin X_2 + \cos X_1 * \cos X_2 * \cos \Delta Y}\right) \tag{2.5}$$

where ΔY is the absolute difference between Y_1 and Y_2 [62].

2.2.6 Comparison

The following example offers a comparison between the presented formulas using a small self developed Java program. It calculates the distances between two points using the Euclidean method, the Law of Cosines, the Haversine and Vincenty's formula. For the first route, two locations near each other were chosen to show the similar results with all methods for short distances. For the second and the third route, long distances were selected to underline the drawbacks of some methods for greater distances.

The first method used for the calculation is the Euclidean distance with the equation shown in 2.1. Since the locations are given as latitude and longitude values, a conversion to Cartesian coordinates is necessary. To convert a point with (latitutde, longitude) to the Cartesian representation (X, Y, Z) the following equations are used, where *rlat* and *rlng* are the radian values of latitutde and longitude and *r* is the earth's radius with 6371 kilometers:

$$\begin{aligned} X &= r * cos(rlat) * cos(rlng) \\ Y &= r * cos(rlat) * sin(rlng) \\ Z &= r * sin(rlng) \end{aligned} \tag{2.6}$$

The second method uses the Law of Cosines for the computation. Followed by the third method which uses the Haversine formula from 2.3, again with the Cartesian representation of the latitude and longitude. The last calculation utilizes Vincenty's formula as implemented in the Java Geodesy Library [15].

For the comparison the following three routes were chosen:

- R1: Stephansdom (48.2083, 16.3728) - Natural History Museum Vienna (48.205177, 16.359799).
- R2: Stephansdom (48.2083, 16.3728) - Christ the Redeemer statue in Rio (-22.95158, -43.210482).
- R3: La Paz in Mexico (24.140281, -110.310677) - Nationalpark Isalo in Madagaskar (-22.487164, 45.276531).

Figure 2.4: Route 2 and 3 displayed on Google Maps

Results

The results are displayed in Table 2.1 as kilometer values and show that for shorter distances all three algorithms are quite accurate and provide nearly identical results. For longer distances though, especially the Euclidean method has a significant discrepancy. The second example route shows a gap of nearly 1000 kilometers between the accurate calculation using Vincenty's formula and the Euclidean approach, whereas the Spherical Law of Cosines and Haversine provide results deviating only by 20 meters from Vincenty's result. For route 3 the gap is even bigger, the Euclidean calculation is off by 5000 kilometers. The percentages of deviation for the Euclidean and the Cosines/Haversine results from Vincenty's outcome are expressed in the columns on the right.

Route	**Euclidean**	**Cosines**	**Haversine**	**Vincenty**	**Deviation Euclidean**	**Deviation Cos / Hav**
R1	1.02	1.02	1.02	1.02	0%	0%
R2	8919.62	9880.42	9880.42	9860.49	9.54%	0.20%
R3	12498.20	17518.51	17518.51	17531.31	28.71%	0.07%

Table 2.1: Comparison of the distance calculation methods (km).

2.3 Positioning Methods

In this section, techniques and methods for locating a user on the earth are discussed. There are different ways of calculating a position, but nearly all of them are based on one of the following three techniques [63]:

1. Trilateration
2. Triangulation
3. Centroid Localization

Trilateration calculates the position of a spot by measuring the distance to at least three separate reference points. If the coordinates of a point and furthermore the distance to the actual spot are known, you can draw a circle around the point disclosing the distance as depicted in Figure 2.5. The spot needs to reside somewhere along this circle. When the coordinates and distance to a second reference point are also known, the distance circle around the second point will coincide with the first circle in two places restricting the actual position to those intersections. Obtaining a third reference point reveals the exact location at the intersection of all three circles [21].

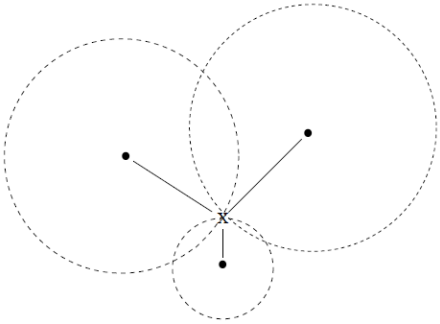

Figure 2.5: Lateration: Determining a position x by using known distances [21].

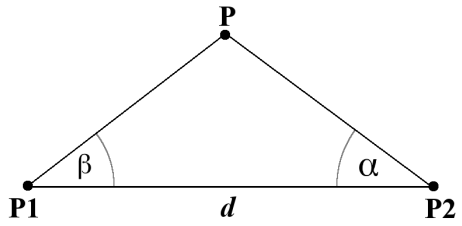

Figure 2.6: Triangulation: Determining a position by using known angles [63].

Triangulation is a technique which uses the trigonometry of triangles to determine the location by measuring the angles to at least two fixed points. The technique requires a known location for the two reference points and the distance between them. Furthermore the two angles α and β as depicted in Figure 2.6 need to be computed. With this information, the two missing edges in the triangle can be determined and the location of the spot can be found at the intersection of those edges [37].

Centroid Localization denotes the technique of determining the position of a spot by obtaining location information of all points nearby and then calculating the arithmetic mean to identify the own position. This method's accuracy depends on the location of the spot within the network and the number of surrounding reference points. The calculation could be imprecise if the spot is located at the edge of the network or if it can only detect a few surrounding points [53].

The choice of the positioning technique depends on the underlying network and its capacities and bandwidth. As depicted in Figure 2.7, positioning methods can be broadly divided into:

1. Handset Based Positioning: The handset measures the data and calculates its own position.
2. Network Based Positioning: The network measures all data and calculates the position of the mobile device.
3. Hybrid Positioning: The device measures the data, then the network calculates the position, since the device usually has less calculation power.

The two most common handset based positioning techniques are GPS and Fingerprinting. Network based methods can be divided into four well-known approaches: TDoA, Cell-ID, ToA and

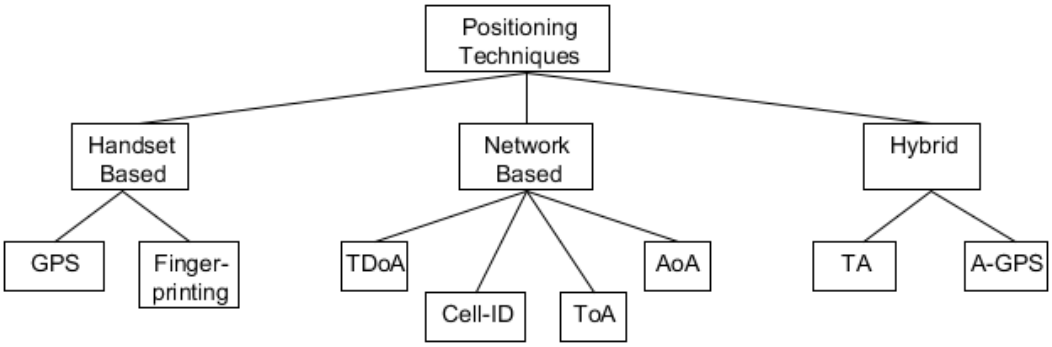

Figure 2.7: Classification of positioning techniques [63].

AoA. Hybrid positioning combines handset and the network based techniques and its basic approaches are TA and A-GPS. All those positioning techniques are explained in detail in the next sections.

2.3.1 Handset Based Positioning

This method is called handset based because the handset itself calculates the position of the user by processing signals from gateways and antennas. It is therefore independent of the cellular network for positioning purposes. The drawback is that the mobile device needs the technologies for determining its position, which makes it more expensive and power consuming. The most common known handset based method is the Global Positioning System (GPS) [2].

GPS

The Global Positioning System was build by the U.S. Department of Defense and is a positioning system based on satellite technology. The satellites transmit radio signals which allow GPS receivers to determine their current position, velocity and time at all hours [67]. GPS consists of three segments: the space, the control and the user segment. The control segment refers of a master control station in Colorado Springs and a system of tracking stations all around the world. The user segment corresponds to the mobile receivers [50].

The space segment has a minimum of 24 active satellites arranged in a manner that at least six of them will be reachable for a receiver. Each satellite sends out broadcast messages with its exact position, status and orbit details (called Ephemeris) and information about all the other satellites (called Almanac) [30]. The GPS receiver measures the transmit time of each message and calculates the distance to the satellites. Then a form of trilateration is used to determine

the current position. The Almanac and Ephemeris information can be saved to enable a quicker startup of the GPS, referred to as a warm start. When the data is not stored, it could take several minutes for a first position fix which is called a cold start. The time it actually takes to find the first position is referred to as time to first fix (TTFF) [65].

The most important advantages of GPS are the availability in all weather conditions, its good coverage and its accuracy and precision. Nevertheless, GPS signals cannot be picked up underground or inside of buildings due to the signal loss from passing through solid objects and can be entirely unavailable during solar storms. Furthermore GPS causes a high power consumption on mobile devices and can also be affected by large buildings or trees [64].

Fingerprinting

Fingerprinting denotes the logging of wireless local area network (WLAN) signals at a specific time and place. It records the MAC address, the name of the router, the service set identifier (SSID), and the received signal strength intensity (RSSI) and stores those distinctive fingerprints in a database, a so-called radio map. Each entry maps a position to a fingerprint. Fingerprinting consists of an offline and an online phase. During the offline phase all RSSI data of the surrounding access points are stored in the radio map, while in the online phase the current RSSI is compared to the map entries. The most similar entry is then assumed to be the current location [25]. The advantage of this method is that the position is determined using existing infrastructure. The hard- and software of the WLAN access points do not need to be changed.

2.3.2 Network Based Positioning

With network based positioning, the infrastructure of the service providers network is used to calculate the position. The mobile device does not require any special hard- or software, the network infrastructure needs to provide all the necessary equipment.

Cell-ID

Cell-ID positioning relies on the network for determining the rough position of the handset by knowing the cell which the device is using. A base transceiver station (BTS or short BS) is a component of a digital network with the purpose of facilitating the communication between network and the mobile devices. Each BS covers a set of cells which are identified by a unique cell identification (Cell-ID). A cluster of cells is called a location area. Each area has its own globally unique local area identity (LAI), a number that identifies the country, network provider and local area code. The BS broadcasts both the LAI and the Cell-ID to its cells and the mobile

devices in it. If the device moves from a cell in one location area to a cell in another location area, the mobile phone performs a location update procedure to inform the network about the change [61].

Since the mobile device can be anywhere in the cell and cells can reach up to several kilometers, the accuracy is related to the size of the cell which serves the mobile device. In [55] Grzegorz Sabak shows that cell sizes depend on the type of the area. Smaller cells are found in city centers to cover the network demand, while in rural areas it is sufficient to use larger cells. According to Grzegorz Sabak this is caused by the fact that cell network operators try to achieve adequate quality when building the infrastructure while at the same time trying to optimize the investment costs.

ToA

Time of arrival (ToA) is based on the arrival time of a signal sent from a mobile device to a base transceiver station. From the time it takes for a signal to travel from A to B the distance between those points can be calculated. To determine the precise location of the mobile device trilateration is used, therefore a collaboration of at least three base stations is required. A precise time synchronization between the stations and the mobile device is very important for an accurate result. Furthermore a time stamp needs to be added to the transmission which enhances the complexity of the signal [69].

TDoA

With the time difference of arrival (TDoA) method, a mobile device sends a signal to surrounding base stations at known locations. Each station records the arrival time of each signal. The current position of the mobile device can then be calculated using the difference of the recorded timestamps of the collaborating base stations. TDoA does not need a time synchronization between the stations and the mobile device, it is sufficient that the base stations use a synchronized clock [69].

AoA

Angle of arrival (AoA) is a positioning method which uses triangulation to determine the position. The mobile device sends out a signal which is received by multiple base stations. The stations then determine the compass direction of the signal. For the positioning, it requires the knowledge of the location of at least two base stations and the distance between them. As shown

in Figure 2.6 the mobile device P sends the signal which is received by the base stations P1 and P2. The stations can then calculate the AoA using the known reference distance d [7].

2.3.3 Hybrid Positioning

Hybrid positioning methods aim to takes advantage of combining handset based and the network based methods. This combination attempts to optimize the accuracy and minimize the time it takes for the location determination.

Enhanced Cell-ID (TA)

In GSM networks, the timing advance (TA) value corresponds to the time it takes for a signal from the mobile device to reach the base station (BS). The TA parameter is only available when then mobile device is in calling mode, which makes it impossible to use TA as a standalone method. However in combination with Cell-ID the TA parameter helps to reduce the location to a smaller area and therefore increases the accuracy. This method is called Enhanced Cell-ID [11].

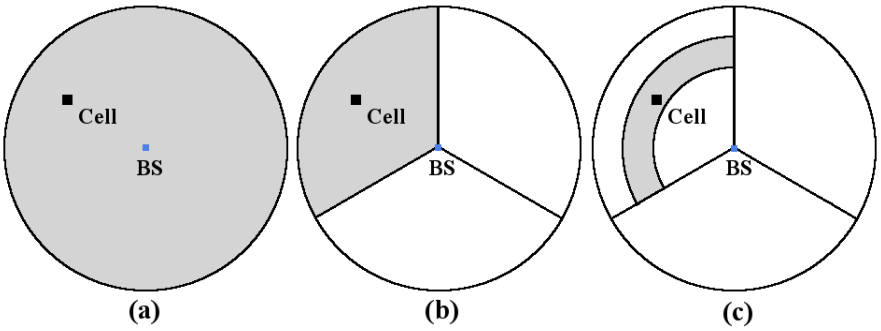

Figure 2.8: (a) Cell-ID (b) Cell-ID and three-directional antenna (c) Cell-ID and TA [9]

Figure 2.8 highlights the advantages of the additional parameters. In the left Figure the position of the mobile device can be anywhere in the gray area surrounding the BS. In urban areas where the BSs are located closer to each other, it is more common that they have three-directional antennas instead of omni-directional ones. This enables a station to cover three cells, also called a sector. Therefore the mobile device can be located more accurately, as shown in the middle picture. Adding the TA parameter to the calculation leads to the most accurate location determination as depicted in the right illustration [9].

A-GPS

Handset based positioning with GPS has a high accuracy, but it takes a relatively long time for a position fix and is not available indoors. To improve this drawbacks, a hybrid approach can be made by merging GPS with a network based technique. The combination is then referred to as Assisted Global Positioning System (A-GPS) [9]. A standalone GPS needs to search for visible satellites and download and decode the orbital information to calculate the current position, which can take a long time. With A-GPS, the network operator provides a server that downloads the necessary data from the satellites and stores it. The data can then be obtained from this server, which speeds up the startup and decreases the power consumption. Nevertheless for A-GPS a significant hardware investment at the mobile device and the network level is required [39].

2.3.4 Comparison

Table 2.2 shows a direct comparison of the positioning techniques presented in the previous sections. The considered metrics are:

1. Accuracy: defines how far off the determined position is to the actual location [63] [32].
2. Reliability: points out the ratio of successful positioning out of all attempts made [2].
3. Latency: refers to the time from the start to the first location measurement [2] [31].
4. Drawbacks: point out some shortcomings of the techniques.

Technology	**Accuracy**	**Reliability**	**Latency**	**Drawbacks**
Handset based				
GPS	5m-50m	high	< 60s	High power consumption, works only outdoors, high TTFF
Network based				
Cell-ID	10m-35km	medium	< 10s	Low accuracy
ToA	100m-400m	medium	< 10s	Needs precise time synchronization between mobile device and the BS, only medium accuracy
TDoA	50m-150m	medium	< 10s	Needs precise time synchronization between the base stations
AoA	50m-150m	medium	< 10s	High investment costs

Hybrid				
Cell-ID & TA	100m-550m	high	< 5s	Only medium accuracy
A-GPS	3m-10m	medium	< 10s	High investment costs

Table 2.2: Comparison of the positioning methods.

Every positioning technique has its benefits and drawbacks, so the right choice depends on the different factors of the actual scenario. Such factors can include the required accuracy level, the power consumption, the current location (indoors or outside) or the desired time to acquire the position. For the ATG a combined approach between Cell-ID, GPS and fingerprinting is used which is provided by Google's Location API. For detailled information on the implementation please refer to Section 5.3.2.

CHAPTER 3

Natural Language Processing

Natural language processing (NLP) is a scientific field which analyzes the connection between human language and computers. One of the earliest approaches in NLP was to count the word occurrences within documents. Since then, many research topics have evolved. Major tasks include named entity recognition, question answering, document retrieval and topic recognition. This chapter attends to the subtasks information retrieval (IR), text summarization, document classification and speech processing since those are the techniques that were used during the implementation of the Automatic Tourist Guide (ATG).

3.1 Information Retrieval

Information retrieval (IR) describes the task of finding and retrieving relevant information or documents containing the desired knowledge within a large collection. The difficulty of this task lies in connecting the search with the corresponding information. A query can be formalized using different terms or languages and the information can be found in different media (text, images, video or audio recording) [47].

Two common areas of application for IR are media searches and search engines. They both have different strengths and challenges. Media searches focus on image, music or video retrieval. Users want to retrieve images based on colors, shapes, textures or abstract concepts. Music retrieval deals with audio recognition based on samples (for example Shazam[1]) or recommenda-

[1] http://www.shazam.com/, last accessed 26.01.2015

tion systems (like Spotify[2]). The area of video retrieval is important to media and news agencies as well as to people who want to manage their own video libraries or browse through a large video on demand selection (for example Netflix[3]). Search engines are another important application of the IR technique. They use the search queries to mine data from databases, the Web or other large text collections. To suit this broad range of application, different models exist to represent the documents.

3.1.1 Models

Information retrieval does not provide concrete answers to given queries, instead it returns the location of text or documents which might include the desired information. In order to obtain the relevant documents, different IR models are available to suit the specific needs. The two most influential ones are [19]:

- The boolean model
- The vector space model

The boolean model is an exact match model where documents are either returned if they are relevant to the search or they are discarded. The vector space model returns all documents in a ranked order which indicates the relevance.

The boolean model

The boolean model was one of the first retrieval models and is based on boolean logic using the operators AND, OR and NOT. The documents are indexed with a set of terms which reflect their content, like tags or keywords. The result of the search depends on whether the indexes satisfy a given query which can contain search terms and operators [17].

Figure 3.1 shows the impact of combining search terms with operators. On the left, the query consists of the two terms *rain* and *sunshine*, combined by the AND operator. This search only returns documents containing both terms. In the middle figure, *rain* and *sunshine* are combined by an OR resulting in the return of all documents containing either one of the terms. The right search delivers all documents which do not contain the word *rain*.

The advantage of this model is that it is easy to implement, the user has control over the search and the result is predictable. Despite these benefits, it is not always easy to translate a search

[2] http://www.spotify.com/, last accessed 26.01.2015
[3] https://www.netflix.com/at/, last accessed 26.01.2005

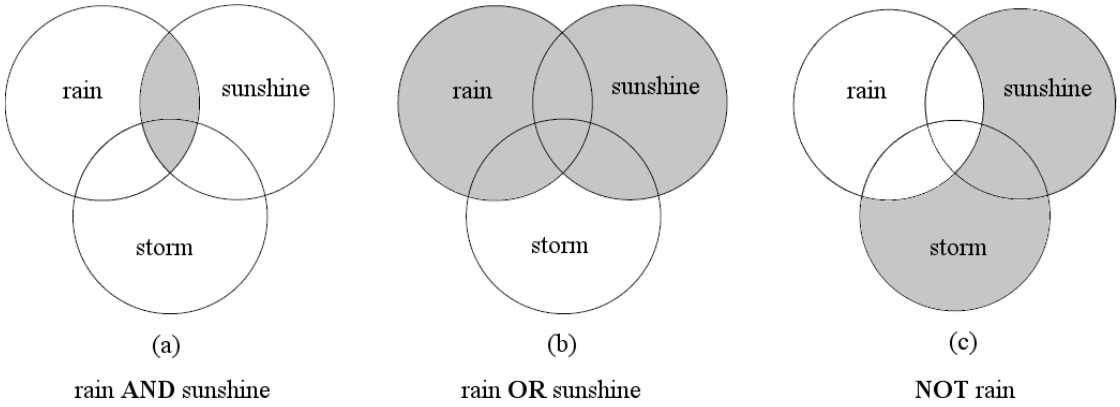

Figure 3.1: Boolean model search terms combined with operators [17].

query into boolean expressions. Another drawback is that all the search terms are equally weighted, therefore the user can not specify terms which are more important than others. Additionally, the model tends to return too many (OR) or too few results (AND) and the traditional boolean model does not rank the retrieved documents [17]. The absence of a ranking function in particular is a big problem since ranking is desirable in most situations. For example, if the search query is "*rain OR sunshine OR storm*", documents which include all three of the search terms are supposed to be more relevant than those which satisfy only one of them.

Over the time, the boolean model has been adapted to eliminate those drawbacks. Extended boolean models were introduced to provide ranking for the retrieved documents. P-Norm is one of the most effective extensions which reduces the strictness of the AND and the OR operator by adding a weight parameter to each term [12].

The vector space model

In the vector space model each query or document is represented by a vector and each term or word by a dimension, for example a document d with the words x_1, x_2 and x_3 is represented as $d = (x_1, x_2, x_3)$ in the model. A vector consists of values for each term which can be binary or weighted. The binary value 1 denotes that a term exists in the query or document and 0 states the opposite. The weight value directly corresponds to the importance of the term. There are different ways for computing the weight values. The most common approach is the term frequency inverse document frequency (TF-IDF) weighting. The *tf* value corresponds to the term occurrences in the document. Therefore, if a word occurs frequently, it is supposed to be significant. This value alone is not sufficient since there are words which naturally occur more often than others, for example pronouns, conjunctions or articles. To balance the weight of these

terms, the *idf* value is added. It measures if a term is rare or common amongst all documents. If a word is frequently used in a lot of documents, the *idf* value scales down the overall *tf-idf* weight and vice versa [13].

Since *tf-idf* is an important concept which is used in different fields of natural language processing, it will be explained in detail in this section. Let *t* be a term, *d* a document and *D* the set of all documents. Then *tf(t,d)* denotes the term frequency value, *idf(t,D)* the inverse document frequency and the weight *tf-idf(t,d,D)* is calculated by the multiplication of those two frequencies *tf(t,d) * idf(t,D)* [27]. For the final result, the vector space model computes the similarity values between the query and each document vector and ranks the results according to the obtained *tf-idf* weights.

The following example was created using [28] as main source. Consider a simple set *D* of three documents:

d1 = "morning sunshine"
d2 = "sunshine and rain"
d3 = "stormy morning"

The first step is to compute the *idf* values by dividing the overall number of documents by the number of those containing the word. Since *idf* computes the logarithmically scaled fraction for a document containing the term, the value is obtained by taking the logarithm for the result of the division:

ifd(morning, D) = $log_2(3/2) = 0.585$
ifd(sunshine, D) = $log_2(3/2) = 0.585$
ifd(and, D) = $log_2(3/1) = 1.585$
ifd(rain, D) = $log_2(3/1) = 1.585$
ifd(stormy, D) = $log_2(3/1) = 1.585$

In the second step, the *tf* values for all terms in the collection of documents D are calculated by counting the occurrences of each term in the documents which yields the following vectors (the order for the terms is $morning$, $sunshine$, and, $rain$, $stormy$):

$d_1 = (1, 1, 0, 0, 0)$
$d_2 = (0, 1, 1, 1, 0)$
$d_3 = (1, 0, 0, 0, 1)$

Afterwards, the *tf* values are multiplied by the *idf* weights of the terms.

$d_1 = (0.585, 0.585, 0, 0, 0)$
$d_2 = (0, 0.585, 1.585, 1.585, 0)$
$d_3 = (0.585, 0, 0, 0, 1.585)$

Let *q* be the query: "morning rain rain". Calculating the *tf-idf* values of the query terms yields the following results:

$q = (0.2925, 0, 0, 1.585, 0)$.

The similarity value of document d_j and query q can finally be obtained by the following inner product [20]:

$$\text{similarity}(d_j, q) = \frac{\sum_{i=1}^{N} w_{i,j} w_{i,q}}{\sqrt{\sum_{i=1}^{N} w_{i,j}^2 * \sum_{i=1}^{N} w_{i,q}^2}} \tag{3.1}$$

Auxiliary calculations for the denominator:

$d_1 : \sqrt{0.585^2 + 0.585^2} = 0.8273$
$d_2 : \sqrt{0.585^2 + 1.585^2 + 1.585^2} = 2.3166$
$d_3 : \sqrt{0.585^2 + 1.585^2} = 1.6895$
$q : \sqrt{0.2925^2 + 1.585^2} = 1.6118$

$similarity(d_1, q) = (0.585 * 0.2925 + 0 + 0 + 0 + 0)/(0.8273 * 1.6118) = 0.1283$
$similarity(d_2, q) = (0 + 0 + 0 + 1.585 * 1.585 + 0)/(2.3166 * 1.6118) = 0.6728$
$similarity(d_3, q) = (0.585 * 0.2925 + 0 + 0 + 0 + 0)/(1.6895 * 1.6118) = 0.0628$

The similarity values indicate the result for each document given the example query and lead to the following ranked order: *d2*, *d1*, *d3*. This outcome shows that *d2* is the most fitting document while the content of *d3* is hardly relevant.

The advantages of the *tf-idf* technique are the ranked results and the weighting of the terms by importance. As a drawback this method assumes that all terms of the query are independent which in practice is mostly not the case.

3.2 Text Summarization

Text summarization refers to the process of taking source text and reducing it to the most essential content. With automatic text summarization (ATS), the summary is produced by a software or an algorithm. Determining which data is relevant is the main challenge in this field. Information that is considered irrelevant will be discarded in the summary since ATS is a sort of compression where information loss is acceptable [44]. This section presents the different summarization methods, the basic workflow to create a summary, and it gives an overview of the evaluation techniques.

3.2.1 Methods

There are two basic approaches for text summarization: extraction-based and abstraction-based. Extraction-based summaries take the original sentences of a document, identify the most important ones and return them in the summary. Abstraction-based summaries interpret the information of the source document and try to create a concise version of it. This approach includes the reformulation of sentences which aims to produce a more natural result than the extraction-based summarization [44].

A further metric which distinguishes summaries is the number of the input sources. The process is called single-document summarization if there is only one input document and multi-document summarization if the input consists of several documents about the same topic. Multi-document summarization is a more complex task because of the possible redundancy of information across the input files.

The ATG uses the Java library Classifier4J[4] for the automatic summarization. This library includes a single-document summarizer that produces an extraction-based summary. For further information about the summarization process in the ATG please refer to Section 5.3.3 in the implementation chapter of this thesis.

3.2.2 Workflow

This section describes the basic workflow of the text summarization process to generate extraction-based summaries. The two general steps are [16]:

- Preprocessing
- Sentence selection

[4] http://classifier4j.sourceforge.net/, last accessed 29.01.2015

Preprocessing

Before a summary can be created, it requires an initial preparation of the text. Figure 3.2 shows the necessary steps. At first, the language of the text needs to be identified. Afterwards, the text is split into smaller segments like sentences or paragraphs. Those segments are further broken down into words or phrases which is called Tokenization. Another step is word normalization which includes stemming and lemmatization.

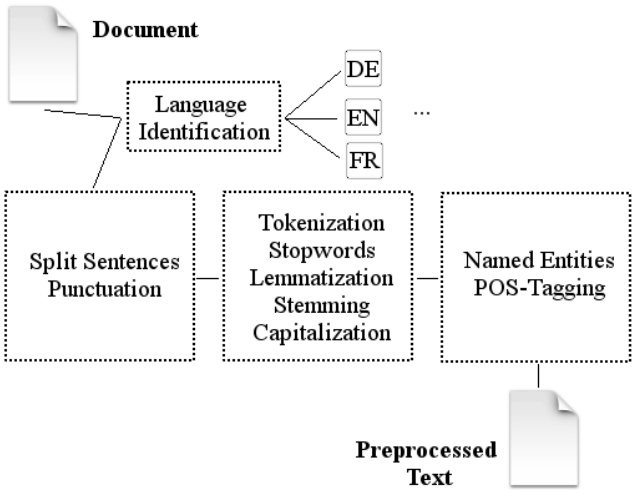

Figure 3.2: Steps for preprocessing a text for the automatic summarization [44].

Stemming refers to the process of reducing a word to its stem by removing the suffix. This step is necessary since different versions of a word can be considered equivalent for the purpose of text summarization. For example, the terms *connection*, *connections*, *connected* and *connecting* can all be stemmed to their base form *connect*. Lemmatization is similar to stemming but it additionally considers the context of the word and it transfers plural or feminine terms into their masculine singular form. For example the base form of *better* is *good* or the French words *suit, sera, eusse, été* are different forms of the verb *être*. These inflections are recognized by lemmatization but not by stemming [44].

In the context of summarization, stop words refer to a set of commonly used words which do not contain a lot of information and are therefore removed from the text. For example, if you have a sentence like “how to tame a dragon“, the words *how*, *to* and *a* are stop words and not considered in the further process.

Determining named entities in the text is another task in the preprocessing phase. It includes the recognition of names and atomic elements in the text like places, people, companies or

expressions of time or quantity. To identify the important sentences of a document, it is crucial to know the topic and the domain of the text. Named entities are seen as the most information dense parts of a document and can be used to define the topic [18].

The last part of the preprocessing step for text summarization is the part-of-speech (POS) tagging which refers to the assignment of words to particular parts of speech like nouns, verbs or adjectives. POS tagging is useful because it is believed that nouns carry the most important information and are more valuable for the summarization than other parts of speech [8].

Sentence selection

After the preprocessing, the document is available as a list of separated sentences. This section presents techniques which are able to determine the appropriate sentences for a summary. Therefore they need to be ranked according to their importance. There are different techniques to determine the value of a sentence. The three most used approaches are: word probability, TF-IDF and the log-likelihood ratio (LLR) [46].

Word probability is a relatively simple approach which uses the frequency of the occurrence of a word as an indicator for its importance. It is computed by dividing the number of times a word appears in the source text by its total number of words. A drawback of this version is that some of the most occurring words in a text might not be the most relevant ones.

TF-IDF weights were introduces to eliminate the drawback of the word probability method. In contrast to word probability which considers all terms equally valuable, the idf value assigns a smaller value to the frequent ones while given rare ones a higher weight. For a precise information on how to calculate the TF-IDF value, please refer to Section 3.1.1.

LLR provides a threshold to mark the words of a text as either important or not. Important words, so called “topic signatures“, are words which occur very often in the input text but appear rarely in other documents. A large background collection is needed to be able to determine the likelihood of a word and to identify the “topic signatures“ [46].

With these three techniques a ranking of the input sentences can be achieved by adding up the weight values for all terms in the sentence. With this ranking, a summarization can be generated by returning the most valuable sentences. Usually, the user provides a numeric value stating how many sentences the summary should have.

3.2.3 Evaluation techniques

An evaluation of summarization methods aims to determine the quality as well as the information content of a generated summary. There are two basic methods: intrinsic and extrinsic evaluations. Intrinsic methods evaluate the summarization itself based on coherence and completeness of the produced summary by comparing it either with human summaries or the source document itself. Extrinsic evaluation techniques determine the usefulness of a summary for performing a certain task, for example using the summary for document classification or question answering [10].

Intrinsic evaluations can be done either with human help or automatically. For human intrinsic evaluations, participants are asked to rate the summary according to its readability and structure. Furthermore the clarity of the text is evaluated, for example pronouns like “he“ or “she“ at the beginning of a sentence have to refer to a noun in a previous sentence. Another step in the evaluation is to check if the summary captured all significant information. However, a human evaluation is slow, expensive and difficult to repeat. Automatic intrinsic evaluations compare the generated summary with an ideal reference summary (created by a human) and score it according to the overlap factor [24]. The most common metric to determine this overlap is ROUGE (recall-oriented understudy for gisting evaluation) which counts the number of overlapping entities like word sequences and word pairs [38].

Extrinsic evaluations concentrate on the usage of summaries for other tasks. For example, in relevance assessments the user judges the relevance of a document for a specific topic solely based on the generated summary. Another example task for extrinsic evaluation is a form of question answering where people evaluate if a question can be answered knowing only the information in the summary.

3.3 Document Classification

Automatic document classification refers to the task of systematically assigning documents to one or more predefined classes or categories. Applications for this technique include spam filtering, authorship and language identification, web page classification or sentiment analysis. Although document classification mostly refers to the analysis of text documents, other sources like images, video or audio files are also possible [40]. The focus of this section is the text classification aspect.

An important part of a document classifier is the training data. Each entry consists of a mapping between a document and a predefined category which is the foundation for the classification. An example training entry is the document "Today it is sunny and the temperature reaches 2°C. A chance of snow showers around noon." and the corresponding category "weather". A text classifier starts with such training data and builds a model which is then used to classify new data.

There are a lot of different machine learning classifiers to approach the document classification task, some of the most well-known ones are [26] [48]:

- Naive Bayes classifier
- Support vector machines
- Decision trees

The td-idf weighting presented in 3.1.1 can also be used for text classification. It defines a weight for each word indicating how significant it is. Afterwards the most valuable words are used to identify the best fitting category. For the ATG, text classification is used in order to assign every section of the Wikipedia text to one of the following categories: history, architecture, geography or sports.

3.3.1 Naive Bayes classifier

The Naive Bayes classifier is based on the Bayes' theorem which states how probability is affected by additional information that will be obtained later. When using the Naive Bayes approach for text classification, the document is seen as a bag of words. This method is relatively simple, there is no need to retain the word order or to analyze the context. The following equation represents the theorem:

$$p(C|D) = \frac{p(D|C) * p(C)}{p(D)} \tag{3.2}$$

whereas C is a category and D refers to a document. Furthermore $p(C)$ states the probability of the category and $p(D)$ the probability of the document. $P(D|C)$ is the probability of the document given the category. With this knowledge $p(C|D)$ can be computed which states the conditional probability of a document to belong to category C [3]. Simplified, this technique separates the feature dimensions and uses given training data to compute the maximum likelihood for a document to be a member of a specific category. The Naive Bayes classifier can accomplish high accuracy and speed for large a amount of data and is often used as a spam filter for e-mail systems.

The following scenario shows a simple example for the usage of the Naive Bayes classifier [33]. Given two classes $female$ and $male$ (c_1 and c_2) and a person p named $Jamie$, we want to compute the gender for this person. Classifying the person is equivalent to the question if it is more probable that p is male or female. A training dataset D with names and the corresponding genders is shown in Table 3.1.

Name	**Gender**
Jamie Lynn Spears	female
Jamie Fox	male
Jamie Lee Curtis	female
Jamie Grace	female
Jamie Oliver	male
Jamie Chung	female
Taylor Swift	female
Taylor Kinney	male
Taylor Kitsch	male

Table 3.1: Dataset for Naive Bayes example.

In this example dataset we have four $Jamies$ who are in the class $female$ and two who belong to the class $male$. Additionally, we have three more individuals who are not named $Jamie$, one female and two male ones. The Bayes' theorem can be applied on this dataset as follows:

$$p(male|jamie) = \frac{p(\frac{2}{4}) * p(\frac{4}{9})}{p(\frac{6}{9})} = 0.33 \tag{3.3}$$

$$p(female|jamie) = \frac{p(\frac{4}{5}) * p(\frac{5}{9})}{p(\frac{6}{9})} = 0.67 \tag{3.4}$$

Equation 3.3 computes the probability for a person called Jamie to be $male$ given the small sample dataset D. In the numerator, the probability of being called $Jamie$ given that you are $male$ is multiplied by the probability of being $male$. The denominator states how probable it is to be named $Jamie$. Equation 3.4 reveals the probability for person p to be $female$. This simplified example for the usage of the Naive Bayes classifier shows that it is more probable for person p to be $female$, given our dataset.

In practice, the Naive Bayes classifier calculates the probability for every term in the input text to belong to one of the given categories and then multiplies the results for each category to determine the likelihood. The category with the highest score will be declared as the result category.

3.3.2 Support vector machines

Support vector machines (SVM) are used to classify the data in two categories by defining a line that separates them, a so-called hyperplane. This method also requires training data with known categories for each entry. A data entry is represented by a vector in the vector space. Using this data, the machine can construct a hyperplane which separates the training objects into the two categories by finding the line which is as far away as possible from all the points. Additionally, the margin to the points which are closest to the hyperplane is maximized. A new text document can be classified into one of these categories by using its position in the vector space.

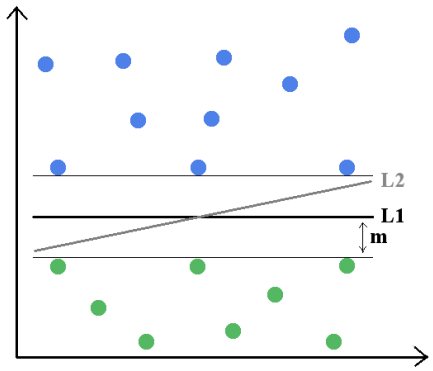

Figure 3.3: Constructing a hyperplane for Support Vector Machines.

Figure 3.3 outlines the difficulties of determining the concrete position of the hyperplane. The blue dots represent data entries belonging to category A and the green dots the entries for category B. There are different possibilities to separate the two categories, for example the line L2. L1 shows the only way to separate them where the margin m is maximized.

Since a hyperplane cannot be bent, a clear separation is only possible if the vectors are linearly separable. In practice this is not the case. There are two approaches to solve this scenario. The first method is to introduce slack variables to allow points on the wrong side of the hyperplane [5]. The second approach is to use the kernel trick to achieve separability. With this technique vectors which violate the separation can be mapped to a higher dimensional space [41]. Figure 3.4 outlines the basic principle. On the left there is a one-dimensional example with two classes: gray and black. Clearly, no straight line can be drawn to separate them. The right picture shows

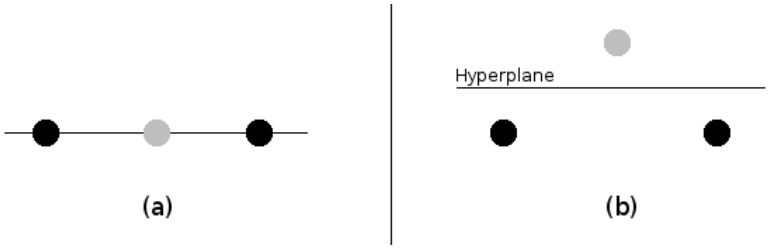

Figure 3.4: Kernel trick for Support Vector Machines.

a two-dimensional space where the gray category has been mapped to a higher dimension as the black vectors. A commonly used kernel which enables this separation is the polynomial kernel.

3.3.3 Decision trees

Classifiers based on decision trees use a tree form to represent the training data. The nodes of the tree are decisions and the leaves are categories. For text documents, such decisions are typically the presence or absence of words within the document. After the creation of the decision tree, the classification process starts from the root node and follows the appropriate branch down to a leaf node which represents the category [56].

For example, the tree for a spam filter could include decision nodes like:

- Was the e-mail sent from a correct domain name?
- Is a subject present?
- Is the recipient's e-mail address present in the *to* or *cc* field?
- Does it include certain words? (Each word gets its own node in the decision tree)

Constructing a good decision tree is the key task for this classification method. There are a number of algorithms to create the tree model. They start by finding an appropriate attribute to partition the data and repeat that step recursively until each data entry belongs to a category. However, if the tree grows too large, it has a risk of overfitting and can not classify new data properly anymore. This situation requires the removal of nodes which are too specific starting from the bottom up. This process is called decision tree pruning [56].

The ATG uses the Apache OpenNLP library[5] for the text classification which is based on the maximum entropy (MaxEnt) principle. MaxEnt is commonly used by document classifiers. It

[5] https://opennlp.apache.org/, last accessed 30.01.2015

works with a “bag of words“ representation of the input text and calculates a value for each word-category pair. The MaxEnt model does not assume the conditional independence of the words, instead it uses the principle that when nothing is known, the likelihood distribution should be as uniform as possible while obeying the given constraints [49].

For example, there is a classification problem with the four different categories: flower, fruit, vegetable and cheese. Without any other knowledge, there are many probability distributions possible, like 20%, 30%, 20%, 30% or 10%, 50%, 10%, 30%. The maximum entropy classifier chooses the distribution which is consistent with the current knowledge and has the highest entropy. It therefore infers a uniform distribution of 25% for all categories. If there is new training data which includes the information that the category flower has a probability of 40%, the classifier infers a distribution of 20% for the other categories.

The basis for a MaxEnt classifier is to identify some features which can be used for the classification, calculate the values for those features amongst the training data and use them as constraints for the probability distribution. The more knowledge the classifier has, the more constraints are introduced and finding an appropriate distribution gets more complex.

3.4 Speech Processing

Speech processing refers to research in the area of speech and includes tasks like speech recognition or synthesis. The former describes the process of converting spoken words into text, while the latter refers to the creation of artificial speech output. A text-to-speech system is able to take text as input and produce computer-generated speech output [35]. The focus of this section is the TTS part of speech processing since the audio output of the ATG is created by Android’s TTS library.

Text-to-speech synthesis faces many challenges. In order to produce a natural sounding voice, it is not sufficient to read each word aloud. The importance is to find the right prosody and intonation. Furthermore, there are linguistic features which are hard to grasp by a TTS synthesizer, like heteronyms and abbreviations. Heteronyms are words which have the same spelling but are pronounced in different ways with different meanings [60]. For example the sentence “The wind was too strong to wind up the sail.“ contains two pronunciations of the word “wind“. Further challenges for TTS systems are numbers as they can represent dates as well as quantity measurements and need to be pronounced accordingly.

3.4.1 Workflow of TTS

The basic workflow for converting text into speech signals contains the basic phases: text processing and synthesis of the speech where each of them includes several subtasks [58]. Figure 3.5 shows a simplified workflow containing the two phases and their subtasks.

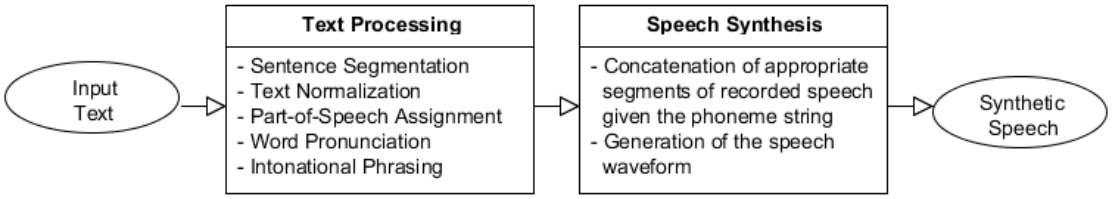

Figure 3.5: Speech synthesis workflow [4].

Sentence segmentation refers to the partitioning of the text into individual sentences. This is not a trivial task because in most languages a period does not necessarily imply the end of a sentence. Text normalization describes the process of expanding abbreviations and numbers into their long form. A POS assignment for each word aids in identifying heteronyms as well as abbreviations since those can also be ambiguous. Word pronunciation deals with the mapping from graphemes to phonemes and with the assignment of the correct intonation, pitch and duration for the words. Furthermore the punctuation marks need to be considered. For example the sentences "'Susan', says Roger, 'is nice'" and "Susan says, 'Roger is nice'" mean two different things depending on the punctuation marks and should be pronounced differently [51].

After the text has been processed and is available as phonemes, the speech output can be generated. This step includes the selection of appropriate segments of recorded speech for the phoneme string and the creation of the speech waveform. It has to be considered that for some speech segments a concatenation of two or more phonemes is necessary, for example for the German segments "ie" or "sch". For the English language about 2900 recorded elements are sufficient to build all phoneme combinations [58].

3.4.2 Android TTS

There are several available TTS engines for the Android platform. This section provides an overview of five TTS applications that can be obtained in the Google Play Store and their strengths and weaknesses. They are analyzed using a small self-developed Android test application which includes nine sentences with different challenges for the TTS engines. The sentences are in German since the ATG is construed for German-speaking regions. The compared engines

are: Google TTS, Samsung TTS, Ivona (Marlene German), CereProc TTS (Gudrun German) and SVOX (Markus German).

For the experiment, nine sentences were chosen and each one includes between one and four challenging terms which are emphasized in bold and explained after the dash. The following sentences were used:

- S1: “Hallo, ich bin Petra und ich mag die **TTS** Technik.“ - This sentence tests if the engine pronounces the capitalized abbreviation TTS correctly, meaning each letter seperately.
- S2: “Das Geschäft am Karlsplatz wurde am **15.10.1950** eröffnet und bereits im Jahre **1960** wieder geschlossen.“ - This sentence includes two dates which should be recognized.
- S3: “Die Abkürzungen **bzw.**, **z.B.**, **usw.** und **s.a.** werden in der deutschen Schrift häufig verwendet.“ - There are four abbreviations present in this sentence and they should be pronounced in their long form.
- S4: “Ich wohne im **12.** Wiener Gemeindebezirk, welcher eine Fläche von **8,16** km^2 besitzt und circa **90.000** Einwohner hat.“ - This sentence includes three numbers in a different context.
- S5: “Leopold **VI** gab einen Bau in Auftrag, obwohl Kaiserin Julia **II** dagegen war, dies ist in Kapitel **VIII** nachzulesen.“ - The first two roman numerals should be pronounced as “Leopold der Sechste“ and “Julia die Zweite“ while the third one should expand to “Kapitel Acht“.
- S6: “Im **3.** Stockwerk befinden sich einige aus dem **17.** Jahrhundert stammende Statuen von König Ludwig und Kaiserin Sissi.“ - The TTS engines should recognize that the period does not denote the end of the sentence.
- S7: “Die Kosten belaufen sich auf **1960** Euro und 90 Cents.“ - This number should not be pronounced the same way as the year in S2; it is not a date.
- S8: “Mögen Sie gerne Wein**?**“ - This sentence verifies if the TTS engines recognize questions and provide a proper intonation.
- S9: “Der Rock ist sehr **modern**, hoffentlich **modern** die Trockenpflaumen nicht.“ - The heteronym “modern“ should be pronounced differently for the two occurences.

	Google	**Samsung**	**Ivona**	**CereProc**	**SVOX**
S1	✓	✓	✓	✓	✓
S2	∼ [1/2]	∼ [1/2]	✓	✓	✓
S3	∼ [1/4]	∼ [1/4]	✓	✓	∼ [3/4]
S4	∼ [2/3]	✓	✓	✓	∼ [2/3]
S5	✓	-	-	∼ [2/3]	∼ [1/3]
S6	-	✓	✓	✓	✓
S7	-	✓	-	✓	✓
S8	-	-	✓	-	-
S9	-	-	-	-	-
Score	8/18 44,44%	9/18 50%	13/18 72,22%	15/18 83,33%	12/18 66,67%

Table 3.2: Comparison of TTS engines.

The results of the evaluation are depicted in Table 3.2 where ✓stands for a completely correct sentence, ∼ means that not all of the difficult text elements were pronounced properly whereas the number in the squared bracket indicates to the correct pronounced elements. The score value is calculated by determining the percentage of challenges each engine got right. Overall there were 18 points to reach, distributed as follows: S1 (1 point), S2 (2 points), S3 (4 point), S4 (3 points), S5 (3 points), S6 (2 points), S7 (1 point), S8 (1 point), S9 (1 point).

The Google TTS engine had problems pronouncing the dates, abbreviations and it did not recognize that the period in S4 and S6 does not mean the end of a sentence which lead to a confusing pause within the text. It was however the only engine which spoke the roman numbers in S5 totally correctly.

Samsung TTS was the most mechanical sounding engine and it struggled with dates. It pronounced 15.10.1950 in S2 completely correct but did not recognize the date 1960 in the same sentence. “Usw.“ was the only abbreviation recognized and all roman numbers were pronounced incorrectly. Nevertheless, it did comprehend the period in S4 and S6 as part of the sentence which lead to a score of 50% for this engine.

The Ivona TTS engine performed well for dates, except for S7 where it read the number like a year declaration. It failed in capturing the roman numbers but it was the only engine which managed to pronounce S8 as a question.

The voice Gudrun of the Cereproc TTS engine was perceived to be the most natural sounding of the five. It was also the one with the best overall score. It recognized all of the abbreviations and dates and even managed to pronounce 1960 right on both occasions (as date and later on as a number).

The SVOX engine did not recognize the abbreviation “s.a.“ and it was the only engine which pronounced the 90.000 in S4 wrong as “90 Punkt null null null“. However, all other numbers were pronounced correctly, all dates were recognized and it managed to capture three of the four abbreviations.

Overall, none of the candidates were able to recognize the two pronunciations of the heteronym “modern“ in S9 and only one had the correct intonation for the question in S8. Although the Google TTS engine did not score high in this particular scenario, it is the default engine for the ATG since it is already pre-installed on a lot of Android devices and most people do not know that they can change their TTS engine. Additionally, the produced speech is more natural than the one of the Samsung engine. For users who want to use their own engine, the ATG provides an option which enables the usage of an arbitrary TTS engine and settings.

CHAPTER 4

Design

The Automatic Tourist Guide (ATG) is an application for the Android platform which offers sightseeing information to the user. Initially, it was created for Vienna but it was expanded to Austria, Germany, Switzerland and Liechtenstein during the implementation phase. This chapter presents the requirements and the use cases for the ATG and the challenges that were faced.

4.1 Requirements

The requirements for the ATG are defined as follows. It shall:

- present POIs to the user which correspond to interesting buildings or structures.
- gather the content for the POIs automatically from the Web.
- display the POIs on a map.
- present the POIs to the user in a visual and an acoustic way.
- offer different versions of the content: a normal version, a summary and a short overview.
- offer different categories for the user to choose from.
- be available offline.
- include an option to pause, resume and stop the audio output.
- include an option to forward and rewind through the sections of the text.

These requirements led to a realization of the project into two systems: a front-end Android application and a back-end which gathers the data from the Web. The back-end system includes

information retrieval and text classification techniques and the front-end system works with text summarization and location-based services.

4.2 Use Cases

This section presents the basic use cases for the front-end and the back-end of the Automatic Tourist Guide. They are described using both a use case diagram and detailed use case descriptions to provide an overview of the capabilities and functions of the ATG.

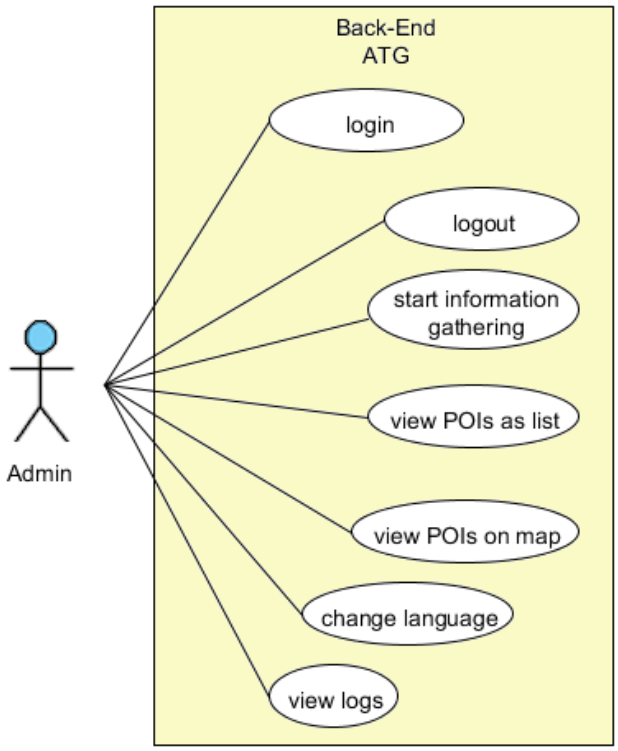

Figure 4.1: Use case diagram for the back-end of the ATG.

The back-end part of the ATG is a web application only accessible for administrators and is henceforth called ATG-Gatherer. Its main task is to fetch the information from Wikipedia. Additionally, it provides log details about the data mining process and an overview of all the gathered POIs. Further information about the implementation can be obtained in Section 5.2.

The font-end part of the ATG is an Android application which supports devices with API level 16 (Jelly Bean) and above. The main component is the automatic guider which presents a POI

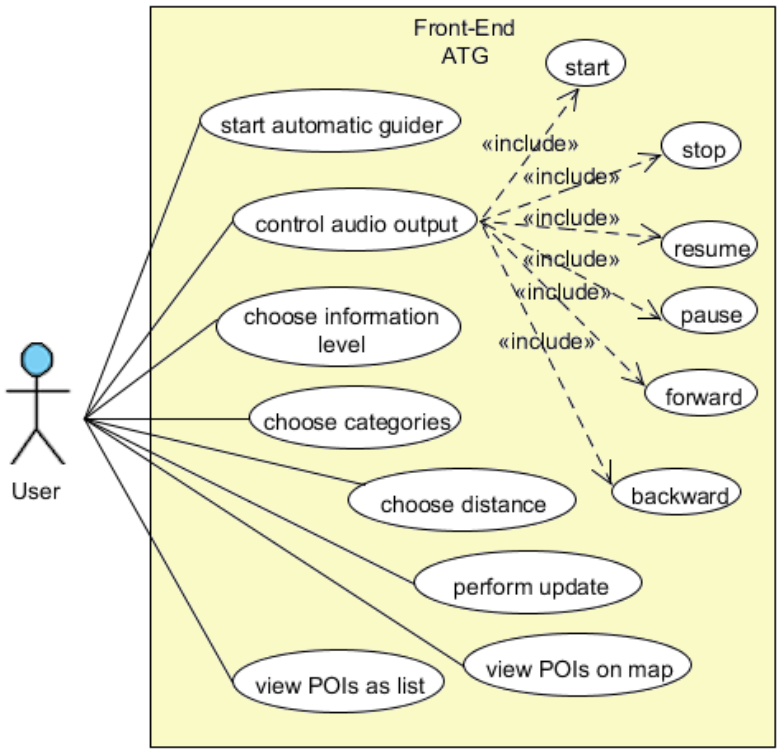

Figure 4.2: Use case diagram for the front-end of the ATG.

in a visual and acoustic way whenever the user comes within a certain distance of its position. Additional features include a map view of all points of interest and a list of the 25 nearest ones. Detailed information about the implementation of the front-end is revealed in Section 5.3.

4.2.1 Use Case Descriptions for the Back-End

Figure 4.1 depicts the central use cases for the ATG-Gatherer. The only actor is the administrator since users are not allowed to access the back-end system. The web application aids in starting the gathering process and choosing the country for which the information should be fetched. At the moment, there are four different countries available: Austria, Germany, Switzerland and Liechtenstein. An administrator can start the gathering process for a specific country and watch the progress output in the web-application. The following pages describe all use cases of the back-end in detail.

UC_BE1	Login
Description	An administrator can login to the back-end web application. There are two predefined users in the database which are allowed to access the back-end. After a successful login, all available features can be utilized and the logs can be reviewed.
Preconditions	- Internet connection is active. - The link to the web application has been opened in a Web browser. - Administrator is not logged in.
Postconditions	The administrator is able to use all features of the web application.
Exceptions	The login will not be successful with a wrong username or password. A proper error message is displayed.

UC_BE2	Logout
Description	An administrator can log out of the back-end web application.
Preconditions	- Internet connection is active. - Administrator is logged in.
Postconditions	A secure logout is performed and the session is destroyed.
Exceptions	-

UC_BE3	Start Information Gathering
Description	The ATG-Gatherer mines Wikipedia for all articles given a main category. It automatically fetches the name of the POI, the coordinates and the content and saves it to a database. All Wikipedia articles which do not have coordinates are discarded. During the process a visual log is provided to monitor the progress or possible errors.
Preconditions	- Internet connection is active. - Internet connection remains active during the process. - Administrator is logged in.
Postconditions	The database is filled with English and German POIs for the main category.
Exceptions	If during the gathering process the Internet connection drops, the ATG-Gatherer continues its work, but all POIs for which the content should have been fetched during the downtime are lost.

UC_BE4	View POIs as a List
Description	All POIs from the database are available in a list view which displays the ID of the Wikipedia article, the name, latitude, longitude and the country of the POI. Furthermore, each entry can be selected to obtain the whole text for the chosen point of interest.
Preconditions	- Internet connection is active. - Administrator is logged in. - Database includes POIs.
Postconditions	POIs can be browsed.
Exceptions	If there are no points available in the database, the list is empty.

UC_BE5	View POIs on Map
Description	All POIs from the database are displayed as markers on a map. The map provides different zoom levels and the possibility to switch between satellite and terrain view. Furthermore, each marker on the map can be clicked to obtain the whole text for the chosen point of interest.
Preconditions	- Internet connection is active. - Administrator is logged in. - Database includes POIs.
Postconditions	POIs can be browsed.
Exceptions	If there are no points available in the database, a map without markers is displayed.

UC_BE6	Change Language
Description	The two available languages for the content are English and German. The administrator can switch between them. The GUI is only available in English.
Preconditions	- Internet connection is active. - Administrator is logged in.
Postconditions	The language changes for the current session.
Exceptions	-

UC_BE7	View Logs
Description	The back-end disposes a web service from which the front-end can obtain updates. Each time a request to access this web service is received, the information gets recorded in the database and can be accessed from the administrator.
Preconditions	- Internet connection is active. - Administrator is logged in.
Postconditions	The logs can be browsed.
Exceptions	-

4.2.2 Use Case Descriptions for the Front-End

Figure 4.1 illustrates the central use cases for the Android application. The ATG is available in two languages, English and German and depends on the system language of the device. The two versions do not have the same POIs, since most of the time, the English Wikipedia does not contain as many articles about sights in German-speaking countries as the German version. The application does not require a registration or login and can be used directly after the installation and update progress is finished. The following pages describe all use cases of the front-end in detail.

UC_FE1	Start Automatic Guider
Description	With the automatic guide modus, the user can walk through the city and is informed whenever a POI is passed. The information about the POI is displayed on the screen and read to the user via TTS. Therefore, it is also possible to turn off the smartphone screen and use headphones to listen to the information.
Preconditions	- The “Guider“-Tab has been chosen. - Location services have been enabled.
Postconditions	- Location-based information about a POI is presented to the user. - The user’s location is updated every 30 seconds.
Exceptions	If the user turns off the location services while the automatic guide modus is still active, the location will not be updated and no new POIs can be presented.

UC_FE2	Control Audio Output
Description	A control over the audio output provides the user with a start, stop, pause and resume option and a jump to the previous or the next section, if there is any.
Preconditions	Automatic guide modus is active or the detail POI view has been chosen.
Postconditions	The audio output is manipulated accordingly.
Exceptions	-

UC_FE3	Control Audio Output - Start
Description	The user can start the audio output.
Preconditions	- The audio output has been stopped. - The POI detail view has been chosen (the automatic guide modus starts the audio output per default whereas in the POI detail view, the user can initiate the process himself).
Postconditions	The TTS engine starts to read the content of the POI to the user.
Exceptions	-

UC_FE4	Control Audio Output - Stop
Description	The user can stop the audio output.
Preconditions	The audio output has been started.
Postconditions	The TTS engine stops to read the content to the user and the POI is marked as visited. If started again, the guider will continue with the next point of interest.
Exceptions	-

UC_FE5	Control Audio Output - Pause
Description	The user can pause the audio output.
Preconditions	The audio output has been started.
Postconditions	The TTS engine pauses the reading. In the automatic guide mode, this option also pauses the guiding process.
Exceptions	-

UC_FE6	Control Audio Output - Resume
Description	The user can resume the audio output.
Preconditions	The audio output has been paused.
Postconditions	The TTS engine resumes the reading at the beginning of the current sentence instead of directly at the word where the output was paused. This feature helps the user to adjust after the pause and provides a better re-entry in the reading process.
Exceptions	-

UC_FE7	Control Audio Output - Forward
Description	The user can forward the audio output to the next section of the POI.
Preconditions	The audio output has been started and is not currently paused.
Postconditions	- The current section is discarded. - The text-to-speech engine resumes the reading at the beginning of the next section.
Exceptions	If the content of the POI does not contain a further section, a short pop-up is displayed to inform the user. The reading process continues without interruption.

UC_FE8	Control Audio Output - Backward
Description	The user can rewind the audio output to the previous section of the POI.
Preconditions	The audio output has been started and is not currently paused.
Postconditions	- The current section is discarded - The text-to-speech engine rewinds to the beginning of the previous section.
Exceptions	If the reading process is still in the first section of the POI, i.e. there is no previous section, a short pop-up is displayed to inform the user. The reading process continues without interruption.

UC_FE9	Choose Information Level
Description	There are three different levels of the text available. The user can choose between the normal content, a summary and a brief overview of the POI.
Preconditions	The “Settings“-Tab has been chosen.
Postconditions	The information level changes accordingly.
Exceptions	-

UC_FE10	Choose Categories
Description	There are four basic categories available: architecture, history, sports and geography. At least one of them has to be chosen. The ATG will only present those sections of the POI that have been categorized with the selected topics.
Preconditions	The “Settings“-Tab has been chosen.
Postconditions	The selected categories change accordingly.
Exceptions	If the user tries to deselect all of the categories, an explanatory error pop-up is displayed and the categories are reset to initial setup which has all of them enabled.

UC_FE11	Choose Distance
Description	The user can change the distance value for which new POI are presented automatically. This option is realized as a dropdown menu with predefined values. For example, the user has started the automatic guide mode, chosen a distance of 100 meter and the nearest POI is 120 meters away. Then the POI is not yet introduced. It will be presented as soon as the user comes within the chosen distance.
Preconditions	The “Settings“-Tab has been chosen.
Postconditions	The distance is set accordingly.
Exceptions	-

UC_FE12	Perform Update
Description	It is possible for the user to refresh the content of the application, if an update is available.
Preconditions	- The “Update“-Tab has been chosen. - A new update is available. - Internet connection remains active during the process.
Postconditions	- All obtained POIs are saved to the database.
Exceptions	-

UC_FE13	View POIs as List
Description	The 25 nearest POIs from the database are presented in a list view which displays the distance from the POI to the users position, the name and part of the first sentence of the content. Furthermore, each entry can be selected to obtain the detail view of the chosen point of interest.
Preconditions	- The “Nearby“-Tab has been chosen. - Location services have been enabled.
Postconditions	The 25 nearest POIs can be browsed.
Exceptions	If the user has the location services disabled, an alert dialog is displayed which aids in activating them.

UC_FE14	View POIs on the Map
Description	All POIs from the database are displayed as markers on a map. Furthermore, each marker can be selected to obtain the detail view of the chosen point of interest. Additionally, the position of the user is displayed if the location services are enabled. The map supports the ability to scroll, zoom and rotate the view and center it to the current position.
Preconditions	The “Map“-Tab has been chosen.
Postconditions	The POIs can be browsed.
Exceptions	If no POIs are available in the database, an empty map is displayed.

4.3 Challenges

For this master thesis, two separate applications were created from scratch. Each of them required a well thought-out architecture due to the large amount of data that should be handled. During the process of creating the two systems, some challenges arose. The most substantial ones are explained in this section.

Back-End

Starting with the back-end, the most challenging step was to obtain a large list of POIs for a predefined area. The starting point was Vienna but a simple extension to other cities or countries should be possible. The initial thought was that for a POI, at first a name and coordinates are needed and then a Wikipedia search can be performed to look up a page about the POI and get the content.

The first attempt was to use the Google Places application programming interface (API) [1] to obtain a list of interesting points for the city. However, this method had several drawbacks. First of all, Google limits the queries in the free version to 1000 within 24 hours which is not enough to obtain the POIs for a whole country. Additionally, the search possibilities are limited. Google offers three different searches: text, nearby and radar search where only the last two are applicable for ATG-Gatherer's purpose. Both methods search for interesting points within a certain radius of a specific position. The nearby search returns less but more detailed results while the radar search presents more results but they only contain the location and the id of a POI (the name has to be fetched in a further request). For both searches, the maximum is 50.000 meters which could cover Vienna, but not a whole country. Furthermore, the query returns at most 200 of the nearest results and they included a lot of places which do not have a Wikipedia page, for example coffee houses and other shops. Overall, with this limitations the Google Places API did not meet the requirements for the POI gathering process.

The solution was found in the category system of Wikipedia which makes it possible to get all the information directly from Wikipedia without the need for additional sources. The category system groups together pages which have similar topics. Starting from a main category, it is possible to gather all subcategories recursively and retrieve all available pages within this categories. For more information about the implementation of the resulting ATG-Gatherer, please refer to Section 5.2.3.

[1] https://developers.google.com/places/documentation/, last accessed 09.02.2015

Front-end

The main challenge for the front-end, the Android application, was the large amount of data. In the final version of the ATG, the database includes approximately 45.000 POIs for the four main German-speaking countries which requires lazy loading for the graphical user interface (GUI) and an optimized database (indices and bulk inserts). This large data set was also problematic for the use case *UC_FE14* which requires to display all points on a map. A clustering solution is used to maximize the performance. All optimization steps can be found in the implementation chapter of this thesis.

The last mentionable challenge was to provide a control function for the speech output, since Android's TTS API has no pause option. It provides a stop option, but if started again, the engine resumes at the beginning of the text. The solution was to read the text sentence by sentence which made it possible to save the index of the current sentence and resume the output at this position after the engine was stopped.

4.4 User Interface

For the design of the ATG, a separate GUI for both the front-end and the back-end was created. The interface for the ATG-Gatherer is simple and rather serves a functional purpose. This section presents the user interface for both applications with focus on the design of the Android application since this is the front-end that is presented to the users.

4.4.1 GUI of the Back-End

The most interesting view of the back-end is the map where all POIs are displayed. Figure 4.3 shows the map view at a zoom level where all POIs are visible as markers on the map. Figure 4.4 shows the map with a higher zoom level and the detail view for a selected POI.

In Figure 4.5, the process of gathering the information from Wikipedia has been started for the country Liechtenstein and the corresponding output is shown. As displayed in the log, the process starts with mining the English Wikipedia for all available sub categories to the main category "Buildings and structures in Liechtenstein". Afterwards, all pages are fetched for the retrieved categories and if a page has geographic coordinates, the content is extracted and the result is saved in the database. Subsequently, the process starts over to fetch the German content.

Figure 4.3: Back-end map view with all POIs.

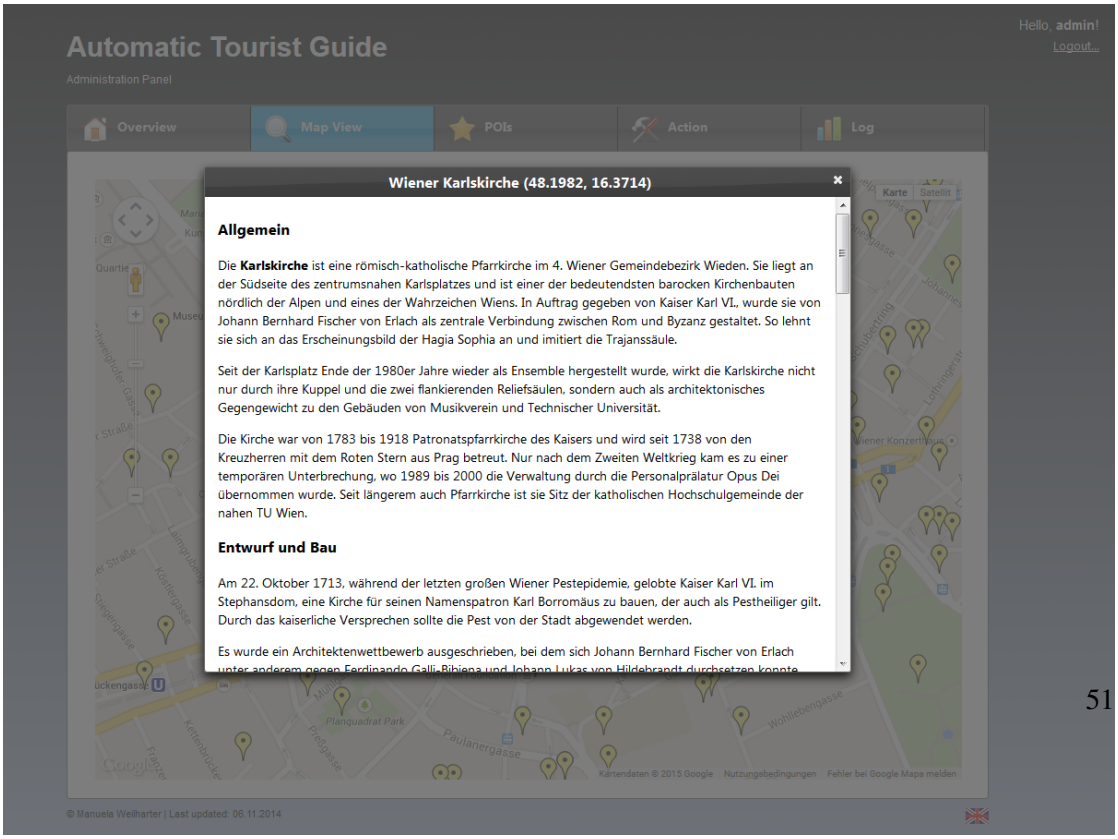

Figure 4.4: Back-end detail view of a POI.

Since Liechtenstein does not have many Wikipedia entries, the gathering process is quite fast. The speed depends largely on the amount of articles for the specific country and can take between two and seven hours with the current implementation. A major improvement of the process could be reached if it becomes possible to query the coordinates and the content at once, since this would cut the number of queries in half. At the moment, the system used to query Wikipedia does not support this feature. More details on the implementation of the ATG-Gatherer can be found in Chapter 5.2.3.

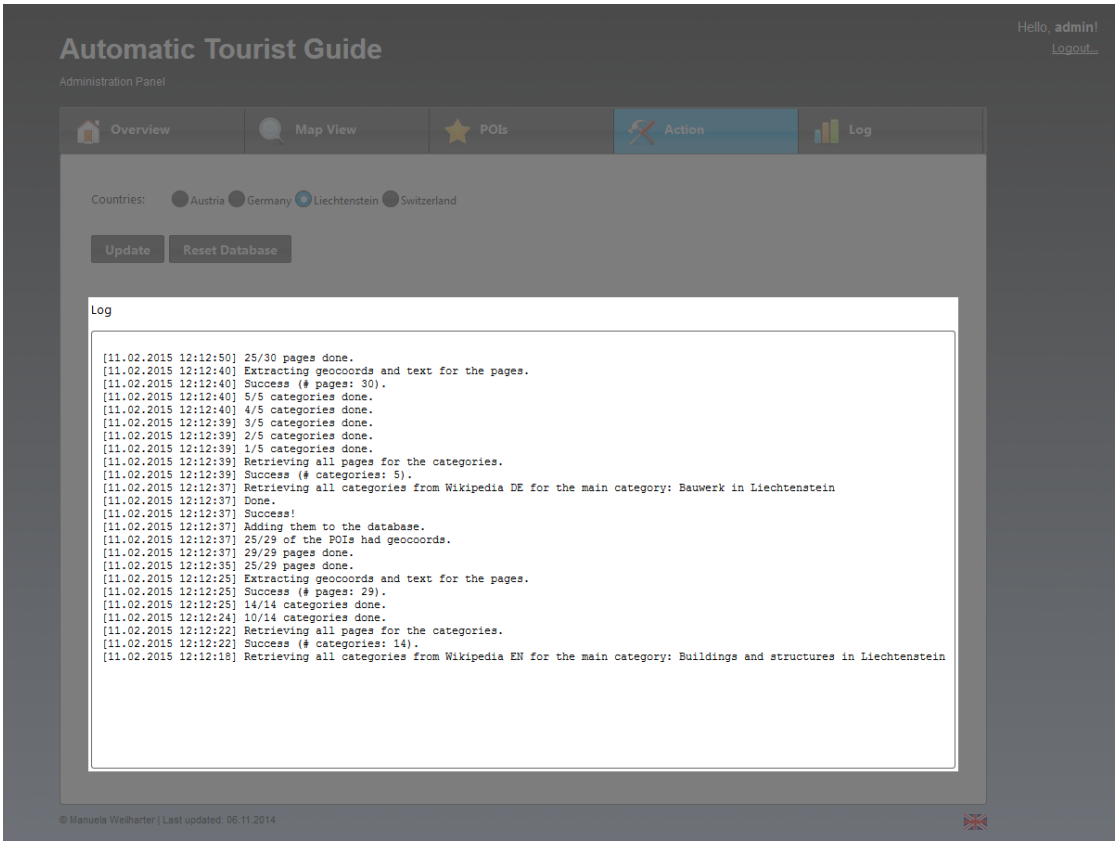

Figure 4.5: Back-end action view with an active gathering in progress.

Overall, the statistics for the back-end database at the moment (11.02.2015) reveal the numbers depicted in Table 4.1. The majority of the POIs are for Germany and the German content version. Austria has only a quarter of the entries and Liechtenstein only contains 38 POIs. The total number of entries in the database adds up to 44823.

Country	# English	# German	Total
Liechtenstein	17	21	38
Switzerland	2668	3357	6025
Austria	1096	6909	8005
Germany	6638	24117	30755
Total	10419	34404	44823

Table 4.1: Statistics for the available POIs.

4.4.2 GUI of the Front-End

The screenshots to outline the front-end GUI were made on a Samsung Galaxy S5 mini which has a display size of 4,5 inch. Figure 4.6 outlines the home screen after the application has been started. The two bars of the guidepost are clickable. A click on the “GUIDE“ button starts the automatic guide modus while a click on “MENU“ opens the side menu shown in Figure 4.7. The side menu can also be opened by clicking on the three stripes left of the application logo or by swiping from the left edge of the screen to the right.

Figure 4.6: Home screen.

Figure 4.7: Side menu.

The side bar menu items have the following functions, from top to bottom:

1. “Guide“: opens the home screen from where the automatic guide modus can be started.
2. “Nearby“: opens the list view showing the 25 nearest POIs.
3. “Map“: opens the map view showing all POIs as markers on a map.
4. “Settings“: opens the settings view where the application can be personalized.
5. “Update“: opens a view from which an update can be performed.
6. “About“: opens a view displaying information about the application.

Figure 4.8 shows a list view where 25 points in the vicinity of the user’s location are displayed. The entries in the list consist of the left side with the actual distance from the user to the POI and the right side with the name and the beginning of the content. An unvisited POI has a blue heading while a visited one is grayed out. Each POI is clickable and leads to the detail view which is depicted in Figure 4.9.

Figure 4.8: Nearby view.

Figure 4.9: POI detail view.

The detail POI view contains a small map where the route to the POI is displayed if the user has an active Internet connection. Below, the actual content is shown including the headings for each section of the text. The sentence which is currently read by the TTS engine is highlighted. At the bottom of the screen are buttons for controlling the TTS output.

The map view of the ATG front-end is presented in Figures 4.10 and 4.11. In Figure 4.10, the standard zoom level is shown which presents all POIs in the vicinity. Each opaque red marker represents an unvisited point of interest while the transparent red markers stand for visited ones. A click on a marker results in a short info frame with the name of the attraction. Furthermore, this info frame is clickable which leads to the detail view of the POI as shown in Figure 4.9. Figure 4.11 shows the map with a higher zoom level and the clustering solution which was used to manage the large amount of markers on the map. The number in the cluster circle represents the number of markers which reside within the clusters radius.

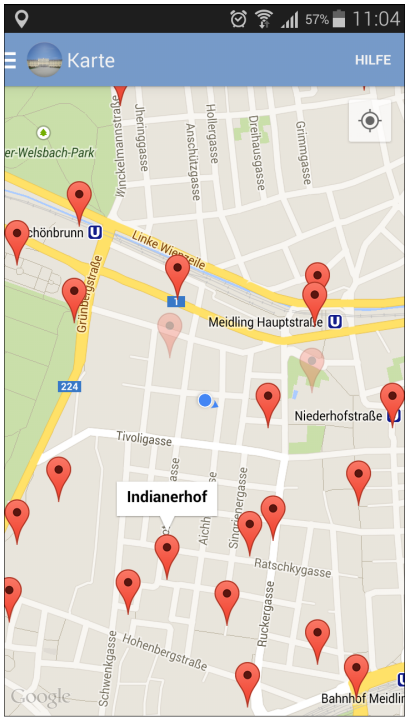

Figure 4.10: Map view.

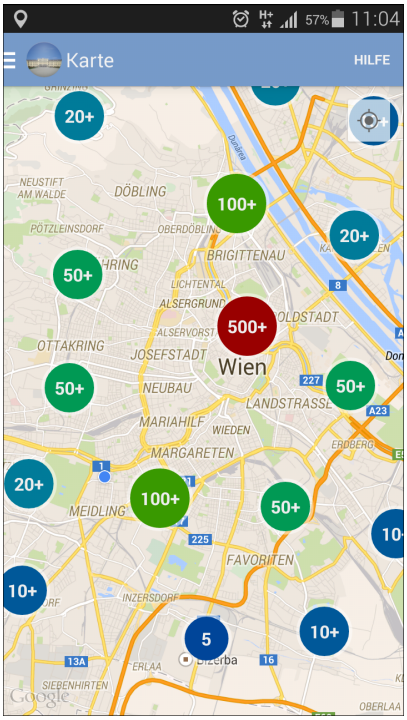

Figure 4.11: Marker clustering.

The last two screenshots show the available settings. Figure 4.12 presents the top half of the options and Figure 4.13 the bottom half. The first option is the category menu where the user can select the categories in which he is interested in (use case: *UC_FE10*). The next option

enables a switch between the information levels: normal, summary and overview (*UC_FE9*). Option three allows to change the distance for the automatic guider (*UC_FE11*). The next three options were no direct requirements for the application but were requested by the testers to enhance the user experience. Option four turns off the automatic highlighting of the current sentence and option five allows to hide the visited POIs. Next, the number of notifications can be limited. A notification is sent whenever the user is in the automatic guide mode listening to the text for a POI while walking by another one. The intention behind the notifications is that the user gets informed about a new attraction without disrupting the current audio output. At last, there is a reset button which enables to flag all POIs as unvisited.

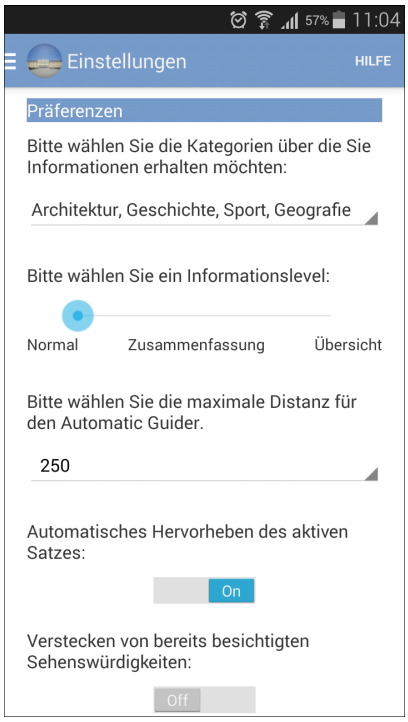

Figure 4.12: Settings part 1.

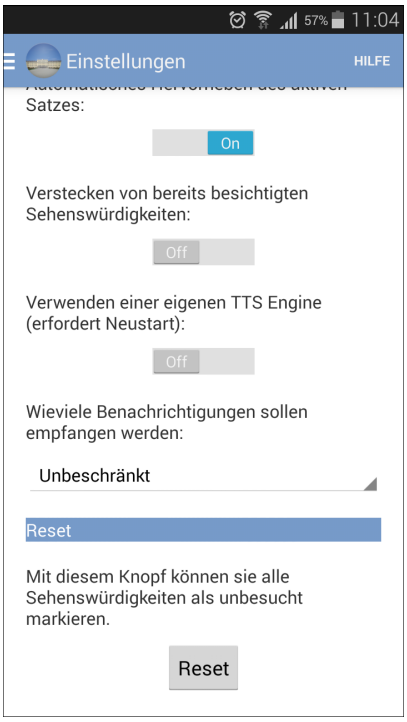

Figure 4.13: Settings part 2.

The two views not captured in a screenshot are the “Update“ and the “About“ view since they do not contain interesting GUI elements. The update fragment consists of a button to start the update function and a short text to explain the procedure. The about fragment reveals the current version of the application and the developer.

Overall, the test users were very pleased with the user interface of the mobile application. The only remark was that the update takes too long. This is due to the fact that about 200 megabytes of data have to be fetched from the web service at the back-end. Increasing the speed of the update might require a different architecture approach and is an improvement opportunity for future versions of the ATG.

CHAPTER 5

Implementation

This chapter describes the technical part of the Automatic Tourist Guide (ATG). It discloses a rough overview of the overall architecture and explains the separation into the two main components: the back-end and the front-end. The project and the database structures are revealed and the most important libraries that were used are introduced. The section for the back-end contains the presentation of the information retrieval solution and the text classification approach. The front-end section deals with location-based positioning and outlines details about the automatic summarization process.

5.1 Architecture

The overall architecture of the ATG consists of two separate systems as shown in the UML deployment diagram in Figure 5.1. The back-end is realized by a web application deployed on an Apache Tomcat server which exposes a web service. The front-end is an Android application which can use a HTTP request to obtain an update from the back-end. Both systems use a relational database to store the data.

5.1.1 Back-end Architecture

The back-end architecture has been realized using the model–view–controller (MVC) design pattern. The MVC pattern assists in defining the architecture of web applications. It provides a separation of concerns for the project and simplifies parallel development. The view component requests data from the model and presents it to the user. The model includes the business logic and encapsulates the state of the application. It provides the data and informs the view about

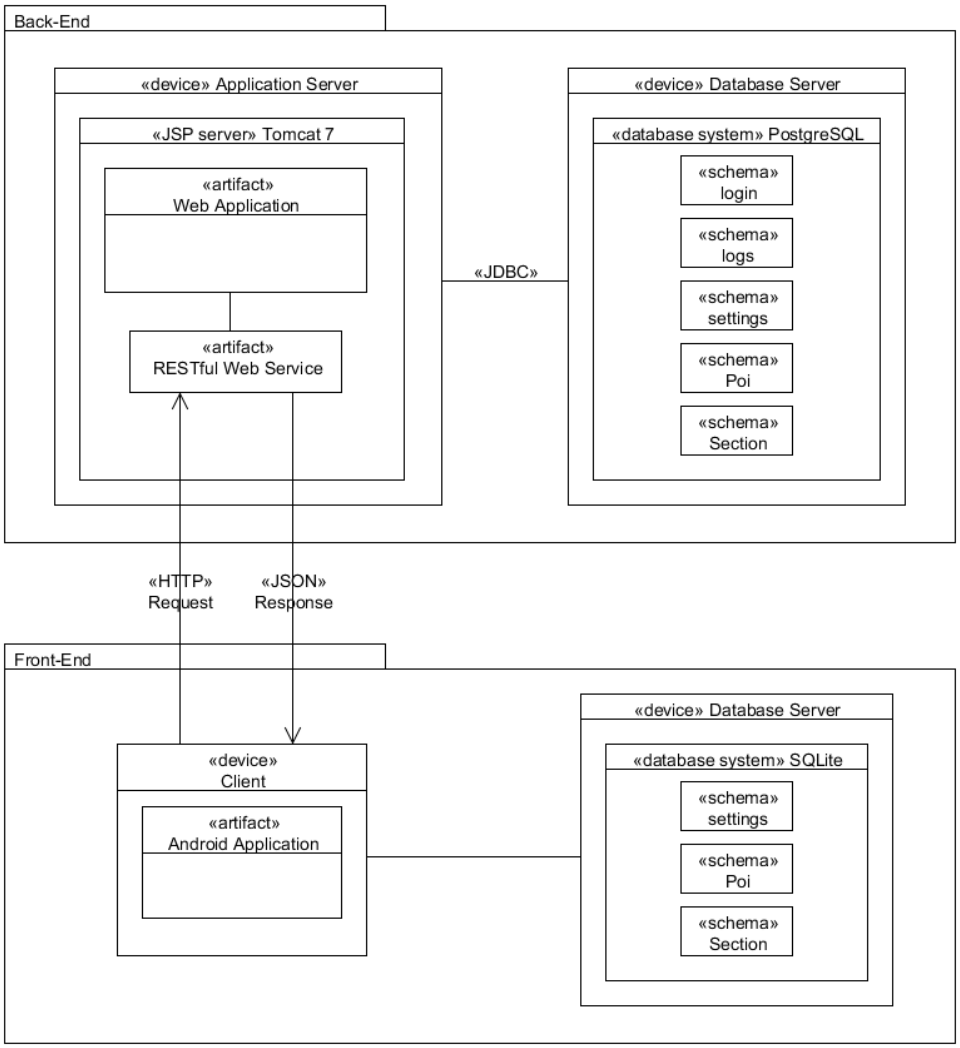

Figure 5.1: Overall deployment diagram for the ATG.

changes. The controller defines the behavior of the application. It is notified by the view about user interactions and updates the model accordingly.

Figure 5.2 shows the package structure of the back-end architecture. The packages are partitioned as follows:

- «entities»: includes the classes that represent the database tables: *poi*, *section* and *log* and two additional helper classes.
- «controller»: includes a controller for each view. The controllers use classes from the

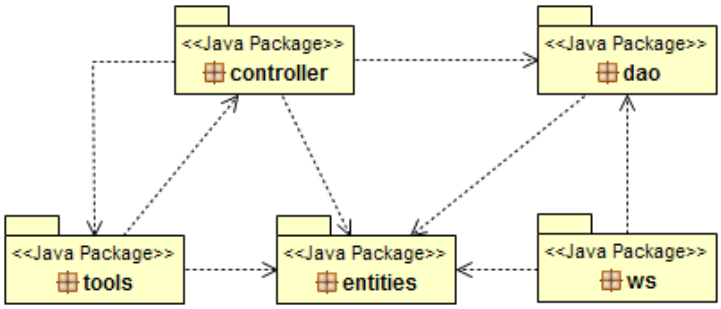

Figure 5.2: Packages diagram for the back-end.

«tools» package to perform specific actions. For example, one tool class is the *NLPHelper* which provides methods for cleaning and formatting the POI content.

- «dao»: includes the data access objects which perform the create, read, update, delete (CRUD) operations for the database. It is used by the controllers and the web service to access the database.
- «ws»: includes the class for the web service.
- «tools»: includes the classes for the Wikipedia gatherer, the text classification and the content cleaner.

The back-end is realized as a JavaServer Faces (JSF) web application. The model component corresponds to the «dao» package and the entities it produces and consumes. The «controller» package represents the controller part of MVC and handles the incoming commands from the views. The view part is not present in the package diagram, since it is located in another branch of the project, the */src/main/webapp* folder. It contains the *index.xhtml* view in the root and the other views in a */secured* folder. This is a security measure to guarantee that only authorized persons can use the functionalities of the web application and no intruder can pass by without a login. A WebFilter[1] is used to realize this task. The source code for this filter can be reviewed in Appendix A.1.

A UML class diagram concludes this section by exposing the overall architecture in Figure 5.3. The interactions among the classes are displayed as arrows where a continuous line indicates that an object was created as instance variable and a dotted line denotes the access within a method. An arrow with a white head refers to inheritance.

[1] http://docs.oracle.com/javaee/6/api/javax/servlet/annotation/WebFilter.html, last accessed 24.02.2015

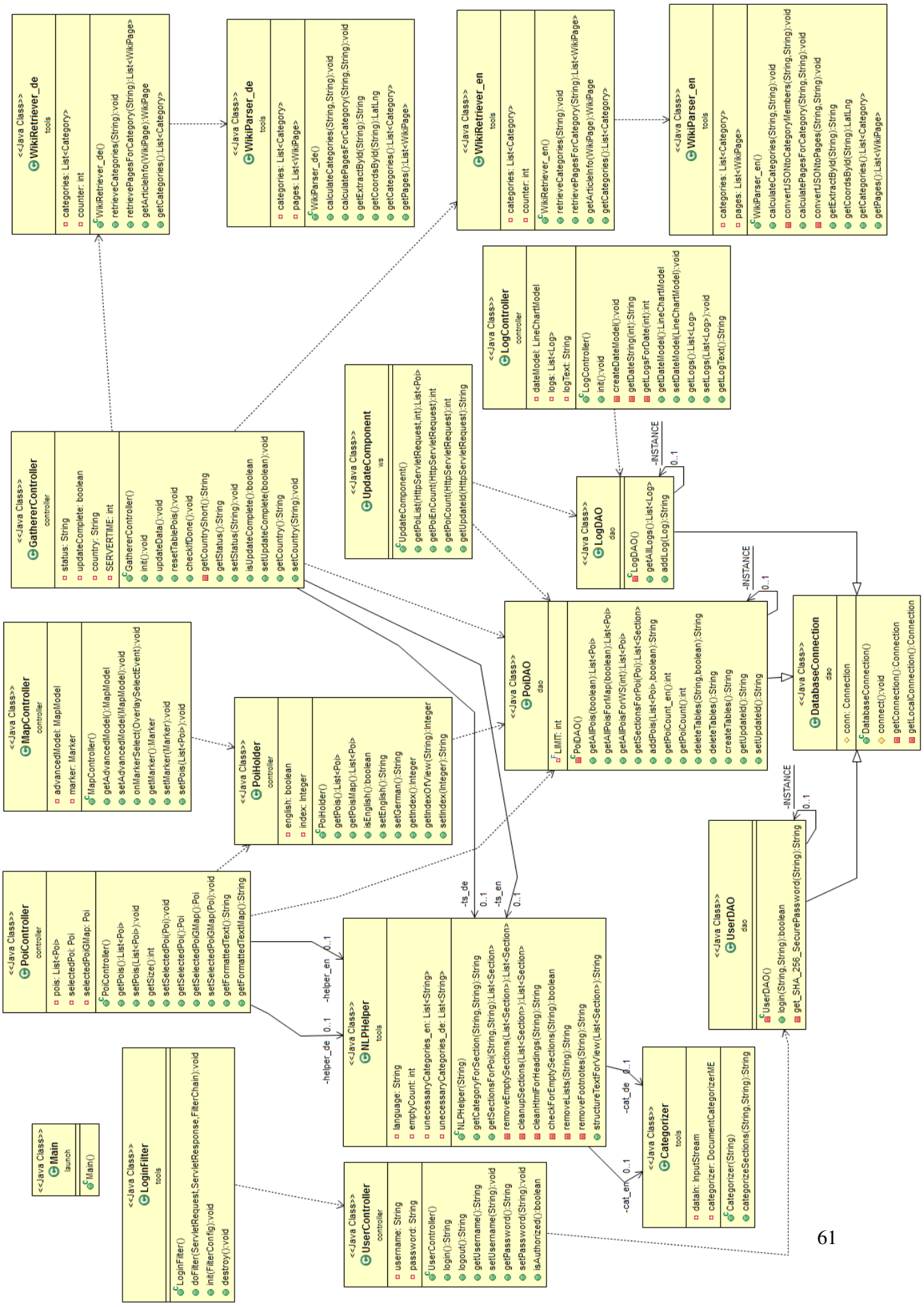

Figure 5.3: Class diagram for the back-end.

5.1.2 Front-end Architecture

The front-end Android application was created using the Eclipse IDE with the ADT Plugin[2] and later migrated to the Android Studio IDE[3] due to Googles announcement that it is now the official integrated development environment (IDE) for Android. The architecture corresponds to a typical Android project using Gradle[4] as build automation system. Figure 5.4 reveals the internal structure of the front-end.

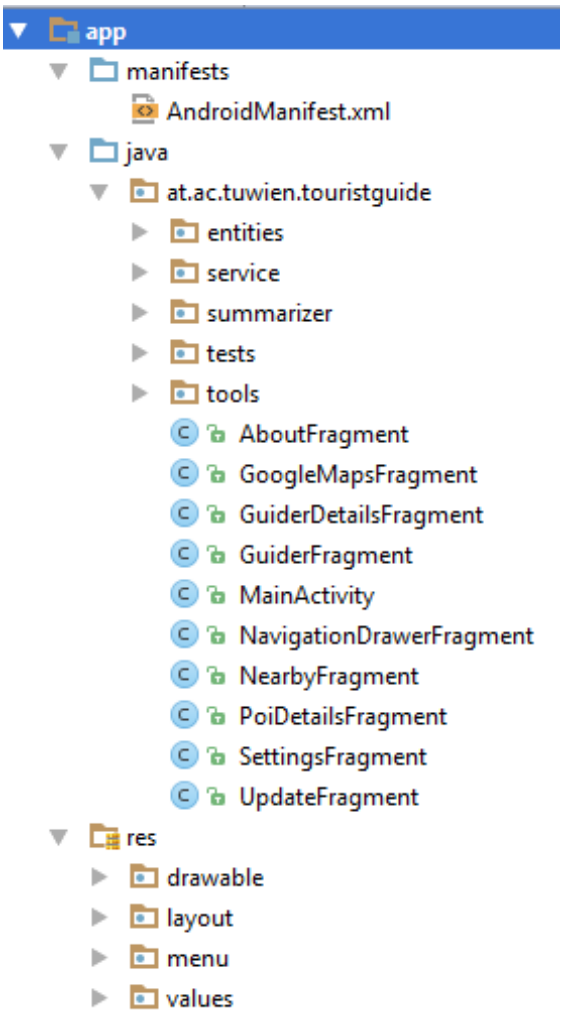

Figure 5.4: Package architecture of the front-end.

[2] http://developer.android.com/tools/sdk/eclipse-adt.html, last accessed 24.02.2015
[3] http://developer.android.com/sdk/index.html, last accessed 24.02.2015
[4] https://gradle.org/, last accessed 24.02.2015

Next to the classes for the view component, the packages inside the Java source folder are partitioned as follows:

- «entities»: includes the classes that represent the database tables: *poi* and *section* and two additional helper classes.
- «service»: includes a location and a TTS service class. They are implemented as Android services which run in the background until the application is closed.
- «summarizer»: includes the necessary classes for the text summarization task.
- «tests»: includes the unit and GUI tests.
- «tools»: includes tools which aid in providing the required functionalities. For example the *UpdateOperation* class which performs the HTTP request to contact the back-end web service and receives the update data.

5.2 Back-End

While the previous section explained the architecture of ATG, this section provides details about the implementation of the back-end. First, the database structure is revealed, followed by a presentation of the libraries that have been used for the implementation. Next, the information gathering process is explained in detail and the methods used to create proper content are explained. The section concludes by revealing details about the web service that is used for the communication between the front-end and the back-end.

5.2.1 Database

The database management system used for the back-end is PostgreSQL[5]. Postgres is an open source object-relational system and provides a JDBC driver to access the database. The database structure is simple and presented in Figure 5.5. The purpose of each table is explained below.

Table Poi: An entry in this table represents the basic information about a Wikipedia article. It includes the name of the POI, the ID of the Wikipedia article and the geographic coordinates. Additionally, the language and the country are stored.

Table Section: A POI has one or multiple sections. They consist of the section header, the text of the Wikipedia page and the category which was determined by the text classifier.

[5] http://www.postgresql.org/, last accessed 24.02.2015

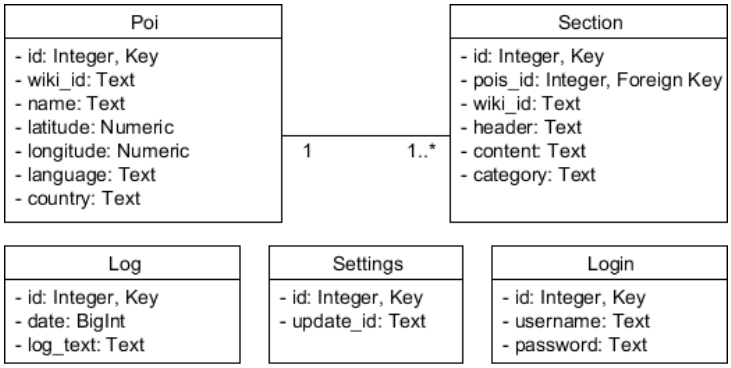

Figure 5.5: Database structure of the back-end.

Table Log: A log entry contains a date and time and the log text. Each time a client accesses the web service, an entry in this table is created.

Table Settings: The settings table is used to save the current ID for the update. The update ID is also saved in the client application. Whenever the front-end initializes an update, this ID is used to check if new data is available.

Table Login: This table contains the authorized users for the web application. The password is saved using a SHA-256 hash function to ensure a secure storage.

Since the two tables holding the POIs and the sections can grow quite large, it was necessary to ensure that queries are executed efficiently. Database indices are used to improve the speed of the data retrieval. Furthermore, only the necessary fields are fetched by a query. For example, the map view of the back-end requires that all POIs are fetched from the database, but it does not require all fields. To place a marker on the map, it is sufficient to know the name and the coordinates of the POI. By fetching only this relevant information, the query speed is maximized. The SQL statements used to create the tables and indices are outlined in Listing 5.1.

```
CREATE TABLE if NOT EXISTS poi (id serial primary key, wiki_id text, name text, latitude
    decimal, longitude decimal, language text, country text);
CREATE TABLE if NOT EXISTS section (id serial primary key, pois_id integer references poi(id)
    ON DELETE CASCADE, header text, content text, category text);
CREATE TABLE if NOT EXISTS settings (id serial primary key, update_id text);
CREATE TABLE if NOT EXISTS login (id serial NOT NULL, username text, password text);
CREATE TABLE if NOT EXISTS log (id serial NOT NULL, date bigint, log_text text);

CREATE INDEX lang ON poi (language);
```

```
CREATE INDEX pois_id ON section (pois_id);
CREATE INDEX cnt ON poi (country);
```

Listing 5.1: CREATE statements for the back-end database.

5.2.2 Libraries

This section presents the most important libraries used for the implementation of the back-end. They are worth mentioning since each of them plays a crucial part in creating an solid back-end information retrieval system.

Primefaces[6] has been used to create the GUI of the back-end. It provides an open source component library for JSF graphical user interfaces. It offers basic elements like buttons, panels and dialogs as well as special features like overlays, exotic menus or different themes.

JSoup[7] is a Java HTML parser which is used to clean up the text. The ATG-Gatherer fetches the content from the Wikipedia sites in a HTML representation. JSoup can then be used to remove elements like lists (OL and UL elements) or to identify each section of the text (each H2 element represents a section header).

Apache OpenNLP[8] is an open source Java library for natural language processing. It supports basic NLP tasks like sentence segmentation, POS tagging and named entity extraction. OpenNLP is used in the back-end to classify the sections of the POIs. Detailed information about the text classifier is provided in Section 5.2.5.

Org.json[9] is used to translate the JavaScript Object Notation (JSON) data into Java classes. The response from the Wikipedia API is received as JSON string and processed using the org.json parser.

5.2.3 The ATG-Gatherer

The ATG-Gatherer is the central part of the back-end. It mines Wikipedia for articles about buildings or structures for a specific country. To accomplish this task, the category system of Wikipedia is used which is explained in the next section. The steps for the gathering process are as follows:

[6] http://primefaces.org/, last accessed 24.02.2015
[7] http://jsoup.org/, last accessed 24.02.2015
[8] https://opennlp.apache.org/, last accessed 24.02.2015
[9] http://www.json.org/java/index.html, last accessed 24.02.2015

1. Select a main category. This can be any category of Wikipedia, but for the ATG, it is "Buildings and structures in *x*" where *x* stands for one of the countries Austria, Germany, Switzerland or Liechtenstein.
2. Recursively gather all subcategories for the main category.
3. Retrieve all pages for the obtained categories. Each page represents a POI.
4. Loop through the pages and query the geographic coordinates. Retrieve the content of the page if coordinates are available and discard it if no coordinates are registered for the site.
5. Divide the content into sections.
6. Remove unnecessary sections and clean up the section content. More information about the cleanup process is revealed in Section 5.2.4.
7. Classify each section.
8. Persist the list of pages in the database.

After these steps, all pages about the given country are available for the ATG. Each page corresponds to a point of interest. This gathering process takes a while because for each page it requires two HTTP requests to obtain a POI since there is no query which can return both the coordinates and the HTML content of the site.

Category System of Wikipedia

A Wikipedia category is a group of articles on related topics. An article can have multiple categories and they are displayed in a box at the bottom of the site. Selecting a category enables the browsing of pages on the same topic or navigate through subcategories. Figure 5.6 shows the category box for the article on the St. Stephan's Cathedral in Vienna.

Figure 5.6: Category box for the St. Stephan's Cathedral.

A click on a category leads to the corresponding category page where all subcategory are listed as well as all articles that have been added to the selected category. For the ATG-Gatherer the main category is not the country itself, it is the subcategory "Buildings and structures in *x*"

where *x* stands for the respective country. This measure is used since the category to a specific country is too broad for the gathering process. It would return a lot of pages which do not have geographic coordinates and would be discarded in the further process.

Wikipedia API

This section provides an overview of the Wikipedia API and the queries used to retrieve the relevant data. The system used is the MediaWiki web API[10] which is a web service providing access to content over HTTP. The next paragraphs describe the queries used to gather all information from Wikipedia.

Retrieving all categories

---

http://en.wikipedia.org/w/api.php?format=json&action=query&list=categorymembers&cmtitle=Category:x&cmtype=subcat&rawcontinue

---

This query is used to recursively obtain all subcategories for a given main category *x*. Starting with the initial category, all members are fetched. For each member, the process starts over until the category has no further subcategories. The used parameters (highlighted in blue) are:

- *format*: indicates the output format of the result. Available formats are JSON, XML and a serialized PHP format.
- *action=query*: is used to indicate that data stored in a wiki is queried.
- *list=categorymembers*: indicates that the query should return all members of a category.
- *cmtitle*: holds the title of the main category.
- *cmtype*: describes the type of the result. For this query the value "subcat" is used to retrieve all child categories.
- *rawcontinue*: allows to obtain further results, if available.

Retrieving all pages

---

http://en.wikipedia.org/w/api.php?format=json&action=query&list=categorymembers&cmtitle=Category:x&cmtype=page&rawcontinue

---

[10] https://www.mediawiki.org/wiki/MediaWiki, last accessed 24.02.2015

The query used to retrieve all pages for a given category *x* has the same parameters as the query for the subcategories. The only difference is that for the *cmtype* parameter the value “page“ is used to return all pages for the category.

Retrieving the content

```
http://en.wikipedia.org/w/api.php?format=json&action=query&pageids=y&prop=coordinates
```

```
http://en.wikipedia.org/w/api.php?format=json&action=query&pageids=y&prop=extracts
```

After the first two steps, there is a list of Wikipedia pages available for further processing. Retrieving the information for each page requires two additional queries. First, the coordinates are retrieved for the given page ID *y*. If coordinates are available, the second query extracts the content of the page in HTML format.

5.2.4 Content Cleanup

The previous section described how the content of a Wikipedia page is extracted. After this extraction, the text is available in a HTML format for further processing. Wikipedia articles might contain sections which are not suitable for the ATG since they do not contain relevant data or are empty due to the parsing process. For example, some Wikipedia pages have a section called “Gallery“ containing pictures about the topic. Those pictures are not returned by the API which results in a section containing only an empty list. Another example are the sections containing the references and literature links which can be found at the bottom of a Wikipedia article. The basic steps for the content cleanup are:

1. Remove irrelevant sections.
   Sections which contain the following headings are removed: “Gallery“, “Notes“, “References“, “Further reading“, “External links“, “Literature“, “See also“, “Sources“ and “Notes and references“.
2. Remove empty sections.
   Additionally to the above mentioned irrelevant sections, some articles have sections containing only pictures. Since images are not parsed by the API, an empty section is returned which is removed in this step.
3. Remove footnotes.
   Wikipedia articles contain footnotes to links or notes. Since in the previous steps those references were removed, the corresponding footnotes must also be deleted.

4. Remove lists.
   Articles can include enumerations of different sorts. For example, the English Wikipedia article about the Vienna University of Technology contains a long list of notable faculty and alumni. Test users found these lists to be uninteresting and even annoying when listening to the information about a POI. Therefore all lists and the corresponding headings and introduction sentences are removed from the content.

After these cleanup steps, the content is partitioned into sections. Therefore each section header of the article and the associated text are identified. The result is a list of sections for each page which is the basis of the text classification process explained in the next section.

5.2.5 Text Classification

The open source library Apache OpenNLP is used for the text classification which assigns every section of the Wikipedia text to one of the predefined categories: history, architecture, geography or sports. The classifier library requires training data which has to be supplied manually. The category with the highest result value is then assigned to the section. In case no category can be assigned to a section, the values are equally weighted to add up to 1. For four categories, this means that each result value corresponds to 0.25. Supplying the classifier with sufficient training data eliminates this problem.

The source code for training and initiating the OpenNLP classifier is available in Appendix A.2. The training data for the classifier needs to be available in a text file in a key-value format. The key corresponds to the category and the value is the text which was classified with the given category. Each data pair needs to be placed in one row and separated with a whitespace. This restraint limits the category to a single word. The categories can be mixed and do not have to be in a special order. An example training file could be:

```
History This is an article which was classified as an historical text.
Architecture This is an article about architecture.
History This is another text about an historical event.
Sports This article contains news about the latest football game.
```

For the training data set, news and Wikipedia articles about the specified categories were used. An expansion to support further categories can easily be accomplished by adding additional key-value pairs to this training data file. They key should refer to the new category. The current training file includes about 50 articles for each category which makes a total of 200 entries.

The text classifier was tested by comparing the actual section headings of the Wikipedia articles with the outcome of the classifier (regarding the used categories architecture, history, sports and geography). For example, the article about the Palmenhaus Schönbrunn contains a section about history and one about architecture. The determined outcome of the OpenNLP classifier was compared with the actual section header to check whether the appropriate category was identified. New training data was added to the classifier until the evaluation returned a match above 90%.

5.2.6 Web service

The communication between the front-end and the back-end is realized using a RESTful web service. The JAX-RS reference implementation Jersey[11] was used to send the data over HTTP. The available methods are:

- *String getUpdateId()*: returns the update ID which is currently stored in the server. The client can check its own update ID against the servers to ascertain if an update is necessary.
- *int getPoiCount()*: returns the number of all POIs in the servers database.
- *int getPoiEnCount()*: returns the number of all English POIs in the database.
- *List<Poi> getPoiList(int offset)*: returns the complete list of all available points. Since this list can be too large to send it in one piece, the offset parameter enables the segmentation into smaller parts.

The source code for the web service is exposed in Appendix A.3. After the web application is deployed to the Apache Tomcat server, a web service method can be accessed using the URL: <*web application address*>*/rest/updater/*<*method name*> For example, if deployed locally, the update ID can be obtained by a HTTP GET request to *http://localhost:8080/rest/updater/getUpdateId*.

5.3 Front-End

This section provides details about the implementation of the Android application. First, the database structure is revealed, followed by a presentation of the positioning techniques that have been used for the implementation. Furthermore, the approach for the automatic summarization is explained.

[11] https://jersey.java.net/, last accessed 24.02.2015

5.3.1 Database

The database management system used for the front-end is SQLite[12] since the Android SDK ships with useful SQLite tools. The database structure is presented in Figure 5.7. It is similar to the back-end and contains the same table setup for the sections. The *poi* table has an additionally value to indicate if a POI has been marked as visited since this information is required in the Android application.

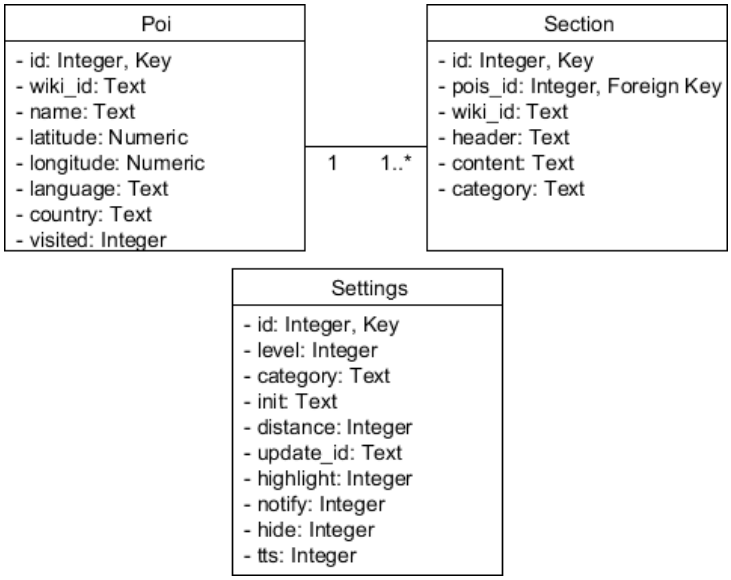

Figure 5.7: Database structure of the front-end.

The *settings* table holds all adjustments which the user has chosen to individualize the application. The columns indicate the following:

- *id*: The primary key used to obtain the current settings.
- *level*: Indicates the currently chosen text level. 0 corresponds to the normal text version, 1 to the summarization and 2 to a brief overview.
- *category*: Indicates the currently chosen categories. The selected categories are saved as a string separated by a whitespace.
- *init*: The value indicates if the application has been opened before. It is used to change the start up routine of the application. When opened for the first time, the application attempts an initial update to obtain the POI. If the update fails, the user can restart it manually using the corresponding menu entry.

[12] http://www.sqlite.org/, last accessed 24.02.2015

- *distance*: Indicates the notification distance which the user has currently chosen in meters.
- *update_id*: This value holds the ID of the last successful update and is used to verify if new updates are available.
- *highlight*: Indicates if the user has chosen to enable or disable the option to highlight the current sentence that is being read from the TTS engine.
- *notify*: This value indicates if the user wants to receive notifications.
- *hide*: Indicates if the user wants to hide the POIs that were already visited.
- *tts*: The last value indicates if the user wants to use the standard TTS engine or the one he has chosen in the "speech and input" settings of his device.

The biggest challenge in the Android development is the handling of the large amount of data. It is not advised to fetch all data of all POIs from the database since this operation can lead to an "out-of-memory" error and a shut down of the application. The solution is to fetch only the necessary fields, the ID, the name and the coordinates. Excluding the content for the POIs reduces the memory usage to a tolerable amount. Nevertheless, even a query for those necessary fields for a couple of thousand POIs can take up to 10 seconds. Executing this operation every time a user opens a view which requires all database rows to be considered would cause lag in the GUI which is not desirable. The solution is to fetch the POIs at application startup and store them in a *POIHolder* class to be available globally. This operation happens in the background while the user sees a splash screen to indicate that the application is being started. This causes a longer startup time, but afterwards the data is quickly accessible. Additionally, database indices are introduced to speed up the queries.

A further challenge is to load all the POIs as markers on to the map. The basic Google Maps Android API requires to run on the UI thread when adding a marker to the map. This results in a freeze of the GUI during this time since no progress bar or info message pop-up can be shown while the GUI thread is busy. The solution involved two improvements. The first step was to add only those markers which are within the bounds of the current viewpoint of the map. For example, in the screenshot in Figure 4.10 only about 25 markers need to be added instead of all 30.000. The next step was to use the Google Maps Android API utility library[13] to enable a clustering of the markers. This is important when the map is zoomed out and the viewpoint gets larger. The source code snippets for this solution are presented in Appendices A.4 to A.6.

[13] http://googlemaps.github.io/android-maps-utils/, last accessed 24.02.2015

5.3.2 Positioning and Distance Calculation

Chapter 2 discusses different techniques for location-based services. In this section the positioning method for the Android application is presented and the technique to calculate the distance between the user's position and a POI is explained.

Positioning

Since the ATG relies on a precise positioning of the user, a combination of LBS techniques was used to retrieve an accurate location. The platform location API in the *android.location* package is the foundation for the location determination.

Google offers two basic methods to receive a location. One uses the network provider (which includes WLAN and Cell-ID positioning) and the other one the GPS provider. Each of them provides a method to receive location updates in a defined interval. Each time such an update occurs, a check is performed to get the best location since the most recently determined one is not necessarily the most accurate. The ATG uses a routine suggested by Google[14] to determine if a new reading is better than the current location fix.

One feature of the Automatic Tourist Guide is that it continues the guiding even when the screen is turned off. Most Android devices have an integrated power saving feature that stops the location services if the user turns off the screen. Therefore, the usage of an Android service for the location implementation was necessary to obtain the desired behavior. A service is a component which is able to perform long-running operations in the background even if the screen is turned off or the user switches to another application. Other classes can bind to this service to interact with it.

Distance Calculation

An accurate distance calculation is an important task for the ATG front-end. The distance value is used to verify if the user comes within a certain distance of a POI and to calculate the 25 nearest attractions. To determine this distance the Java implementation of Vincenty's formula in the Geodesy Library is used [15]. The formula produces very accurate results to within 0.5 millimeters.

[14] http://developer.android.com/guide/topics/location/strategies.html, last accessed 24.02.2015

5.3.3 Automatic Summarization

For the automatic summarization of each section of the content, the Java library *classifier4J* is used with some modifications. The sentence segmentation algorithm of the library splits the sentences at every period which produces confusing summaries since a period does not necessarily indicate the end of a sentence. For example, the sentence "The meeting with the U.S. president was quite memorable." would be partitioned into three sentences. The sentence segmentation algorithm developed for this task still splits the text into sentences using periods as delimiters. But afterwards each sentence is checked if it is a valid sub sentence. For the algorithm, it is not a valid segmentation if:

- The last word of the sentence is part of a list of common words which end with a period but do not indicate a new sentence. For example, titles and abbreviations.
- The last word of the sentence is a number. For example, "Am 9. Oktober", "Im 20. Jahrhundert", "Im 12. Wiener Gemeindebezirk".
- The first word of the sentence starts with a lower case letter. For example, "The U.S. president".

If an incorrect split is detected, the parts of the sentence are reassembled. This improvement of the sentence segmentation is not complete and it might produce false positives. Nevertheless, for the automatic summarization used in the ATG it is less problematic if two sentences are falsely recognized as one and returned in the summary instead of only half of a sentence being included in the result.

CHAPTER 6

Conclusion and Future Work

In the previous chapters, the Automatic Tourist Guide (ATG) has been presented. The techniques used for creating this project were discussed and details about the design and implementation process were revealed. This chapter concludes the work by outlining the results and suggesting improvement and extension possibilities.

6.1 Conclusion

This thesis introduced an application that can be used to guide the user around a city or region. The result of the work is a fully functional system which consists of a back-end web application and a front-end Android application. For the back-end ATG-Gatherer, the goal was to use information retrieval (IR) techniques to obtain and NLP methods to refine and classify the content. For the front-end, a NLP technique is used to summarize the text and a location-based positioning of the user is enabled. The findings of the research are presented in Chapter 2 and 3 and helped to identify the most suitable techniques. The investigation led to the usage of *Classifier4J* for the summarization task which is a single-document summarizer producing an extraction-based summary. For the classification part, *OpenNLP* is utilized which relies on a maximum entropy text classifier. Evaluating different positioning techniques completes the research phase and aids in implementing a precise positioning of the user.

Compared to existing mobile tourism applications, the ATG focuses on the automatic guiding which allows the user to walk through the city and automatically get notified whenever she walks past a point of interest. Furthermore, the content is not added manually by writing an article

about each POI, it is gathered automatically from the Web. This enables a simple expansion to further cities. Additionally, the ATG-Gatherer could be used for other purposes, for example to retrieve all articles about Austrian football players or German politicians.

The technologies used for the implementation were carefully chosen to meet current standards. JavaServer Faces (JSF) was used to create the GUI for the back-end and the front-end application is available as an Android Studio project which is the current standard development tool. Relational databases were used to persist the data since they enable an efficient storage for large data sets. Furthermore, a RESTful web service enables the communication between back- and front-end which is a more lightweight approach than using Simple Object Access Protocol (SOAP) for the communication.

The finished application was tested in Vienna and found useful to discover new parts of the city since it does not only include the major tourism attractions but every POI of the city which has a Wikipedia article. For example, it can also be interesting to read about the architecture and history of a metro station or the small park next to it.

6.2 Future Work

The ATG is already a fully functional application, although there is still room for further improvements. This final section of the thesis outlines future work opportunities for this project and provides suggestions on how to achieve them.

One drawback of the current architecture is the slow communication over the RESTful web service. Updating the Android application requires the transmission of approximately 45.000 POIs which takes up to three minutes at the moment. For a mobile application, this time period is too long since the user can not use the application during the update. Although the process was improved during the test phase by showing the current progress to the user, the time span remains the same. The ATG could benefit from research into different architecture approaches and communication methods for accelerating this update process.

Another process that takes a long time is the information gathering from Wikipedia. One reason for this is that for every page, it requires two queries to obtain the necessary information. Finding a method to retrieve all relevant data in one query would result in a significant speed improvement for the ATG-Gatherer. Additionally, it might be considered to use a database dump of Wikipedia for retrieving the data instead of sending the queries in real time. This would enable an offline gathering and could also improve the query speed.

Currently, the ATG supports four different categories for the user to choose from: architecture, history, sports and geography. The categories can easily be extended, though it requires a careful enlargement of the training data. Representative articles for the new categories need to be obtained and added to the training set. Possible new categories might include *religion*, *entertainment*, *news* or *activities*. Additionally, new text levels can be added. Next to the normal text, the summary and the overview, the text could be converted into simple English or a switch between British English and American English could be included.

The back-end web application is currently only accessible for administrators. A further improvement of the ATG could include an extension of the web application to support multiple user groups. All users should be able to log in and watch the map and the list view of the POIs, but only the administrators should be able to start the information gathering process or to view the logs. Especially for tourists, it could be interesting to watch the map view with all the possible attractions to visit.

The Android application could further benefit from adding pictures for the POIs. The ATG-Gatherer already includes an unused method to query the main picture of a Wikipedia article. The challenge with adding pictures is that they need additional space in the database and when sending the content over the web service. A possible solution would be to save the URL of the picture and to load it when an active internet connection is available. Having a picture for the POIs would be an enrichment for the GUI of the front-end.

At last, the ATG is currently only available for mobile devices with the Android operating system. An implementation for iOS would be beneficial to cover the two most used mobile operating systems and to enlarge the user base.

APPENDIX A

Source Code

This section presents source code snippets of the implementation. Appendix A.1 displays the login filter used to ensure a secure login for the web application. Appendix A.2 reveals the implementation of the text classifier and Appendix A.3 shows the back-end web service. At last, Appendix A.4 to A.6 outline the essential code for setting up the map view of the front-end.

Listing A.1: LoginFilter.java

```
1  import java.io.IOException;...
2
3  @WebFilter("/secured/*")
4  public class LoginFilter implements Filter {
5
6      public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
           throws IOException, ServletException {
7          if (((HttpServletRequest) request).getSession().getAttribute(UserController.AUTH_KEY) ==
               null) {
8              ((HttpServletResponse) response).sendRedirect("/index.xhtml");
9          }
10         else {
11             chain.doFilter(request, response);
12         }
13     }
14
15     public void init(FilterConfig config) throws ServletException {}
16     public void destroy() {}
17 }
```

Listing A.2: Categorizer.java

```
1  import java.io.FileInputStream;...
2
3  public class Categorizer {
4
5      private InputStream dataIn;
6      private DocumentCategorizerME categorizer;
7
8      private static final Logger log = Logger.getLogger(Categorizer.class.getName());
9
10     public Categorizer(String language) {
11         try {
12             if (language.equals("en")) {
13                 dataIn = new FileInputStream("categorizer.model.en");
14                 DocumentSampleStream docstream = new DocumentSampleStream(
15                         new PlainTextByLineStream(dataIn, "UTF-8"));
16                 DoccatModel model = DocumentCategorizerME.train("en", docstream);
17                 categorizer = new DocumentCategorizerME(model);
18             }
19
20             else {
21                 dataIn = new FileInputStream("categorizer.model.de");
22                 DocumentSampleStream docstream = new DocumentSampleStream(
23                         new PlainTextByLineStream(dataIn, "UTF-8"));
24                 DoccatModel model = DocumentCategorizerME.train("de", docstream);
25                 categorizer = new DocumentCategorizerME(model);
26             }
27         } catch (IOException e) {
28             log.severe(e.toString());
29         } finally {
30             if (dataIn != null) {
31                 try {
32                     dataIn.close();
33                 } catch (IOException e) {
34                     log.severe(e.toString());
35                 }
36             }
37         }
38     }
39
40
```

```
41      public String categorizeSections(String content, String language) {
42          content = Jsoup.parse(content).text();
43          double[] outcome = categorizer.categorize(content);
44
45          if (StringUtils.countMatches(categorizer.getAllResults(outcome).toString(), "0,25") == 4)
46              return "";
47
48          return categorizer.getBestCategory(outcome);
49      }
50  }
```

Listing A.3: UpdateComponent.java

```
1   import java.util.ArrayList;...
2
3   @Path("/updater")
4   public class UpdateComponent {
5
6       private static final Logger log = Logger.getLogger(UpdateComponent.class.getName());
7
8       @GET
9       @Produces(MediaType.APPLICATION_JSON)
10      @Path("/getPoiList/{offset}")
11      public List<Poi> getPoiList(@Context HttpServletRequest request,
12              @PathParam("offset") int offset) {
13          String host = "";
14          try {
15              host = request.getRemoteHost();
16          } catch (Exception e) {
17              log.severe(e.toString());
18          }
19
20          LogDAO.getInstance().addLog(new Log("WebService Request from " + host,
                System.currentTimeMillis() + (60 * 60 * 1000)));
21          List<Poi> allPois = new ArrayList<Poi>();
22          allPois.addAll(PoiDAO.getInstance().getAllPoisForWS(offset));
23          return allPois;
24      }
```

```
25
26      @GET
27      @Produces(MediaType.APPLICATION_JSON)
28      @Path("/getPoiEnCount")
29      public int getPoiEnCount(@Context HttpServletRequest request) {
30          return PoiDAO.getInstance().getPoiCount_en();
31      }
32
33      @GET
34      @Produces(MediaType.APPLICATION_JSON)
35      @Path("/getPoiCount")
36      public int getPoiCount(@Context HttpServletRequest request) {
37          return PoiDAO.getInstance().getPoiCount();
38      }
39
40      @GET
41      @Produces(MediaType.APPLICATION_JSON)
42      @Path("/getUpdateId")
43      public String getUpdateId(@Context HttpServletRequest request) {
44          return PoiDAO.getInstance().getUpdateId();
45      }
46  }
```

Listing A.4: Setting up the map and the listeners

```
1  if (Locale.getDefault().getLanguage().equals("de")){
2      allPois = PoiHolder.getPois_de();
3  } else {
4      allPois = PoiHolder.getPois_en();
5  }
6
7  googleMap = mapFrag.getMap();
8  mClusterManager = new ClusterManager<PoiMarker>(context, googleMap);
9  mClusterManager.setRenderer(new PoiRenderer(context, googleMap, mClusterManager));
10 googleMap.setOnCameraChangeListener(mClusterManager);
11 googleMap.setOnMarkerClickListener(this);
12
13 googleMap.setOnInfoWindowClickListener(this);
```

```
14 googleMap.clear();
15 googleMap.setMyLocationEnabled(true);
16
17 googleMap.setOnCameraChangeListener(new OnCameraChangeListener() {
18     @Override
19     public void onCameraChange(CameraPosition position) {
20         bounds = googleMap.getProjection().getVisibleRegion().latLngBounds;
21         addMarkers();
22     }
23 });
```

Listing A.5: Adding the markers

```
1  private void addMarkers() {
2      List<PoiMarker> markers = new ArrayList<PoiMarker>();
3
4      mClusterManager.clearItems();
5
6      for (Poi poi : allPois) {
7          if (bounds.contains(poi.getLatLng())) {
8              markers.add(new PoiMarker(poi));
9          }
10     }
11
12     mClusterManager.addItems(markers);
13     mClusterManager.cluster();
14
15     if (lastOpened != null) {
16         lastOpened.showInfoWindow();
17     }
18 }
```

Listing A.6: PoiRenderer.java

```
1  import java.util.HashMap;...
2
3  public class PoiRenderer extends DefaultClusterRenderer<PoiMarker> {
4
5      public static HashMap<Marker, PoiMarker> markerMap = new HashMap<Marker, PoiMarker>();
6
7      public PoiRenderer(Context context, GoogleMap map, ClusterManager<PoiMarker>
           clusterManager) {
8          super(context, map, clusterManager);
9      }
10
11     @Override
12     protected void onBeforeClusterItemRendered(PoiMarker poiMarker, MarkerOptions
           markerOptions) {
13         LatLng coords = poiMarker.getPoi().getLatLng();
14
15         if (poiMarker.getPoi().getVisited() == 0) {
16             markerOptions
17                     .position(coords)
18                     .title(poiMarker.getPoi().getName());
19         } else {
20             markerOptions
21                     .position(coords)
22                     .title(poiMarker.getPoi().getName())
23                     .alpha(0.3f);
24         }
25     }
26
27     @Override
28     protected void onClusterItemRendered(PoiMarker poiMarker, Marker marker) {
29         super.onClusterItemRendered(poiMarker, marker);
30         markerMap.put(marker, poiMarker);
31     }
32
33 }
```

Bibliography

[1] Diaa AbdElminaam, Hatem Abdul Kader, Mohie M. Hadhoud, and Salah El-Sayed. Increase the performance of mobile smartphones using partition and migration of mobile applications to cloud computing. In *International Journal of Electronics and Information Engineering*, pages 34–44, 1984.

[2] Isaac K. Adusei, Kyandoghere Kyamakya, and Klaus Jobmann. Mobile positioning technologies in cellular networks: An evaluation of their performance metrics. In *MILCOM 2002. Proceedings*, volume 2, pages 1239–1244, Oct 2002.

[3] Charu C. Aggarwal. *Data classification: Algorithms and applications*. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series. CRC Press, 2015.

[4] Rudzionis Algimantas, Kastytis Ratkevicius, and Vytautas Rudzionis. *Voice Interactive Systems*, pages 281–296. John Wiley & Sons, Inc., 2008.

[5] Constantin F. Aliferis, Douglas P. Hardin Statnikov, and Alexander Statnikov. *A Gentle Introduction to Support Vector Machines in Biomedicine: Theory and methods*. A gentle introduction to support vector machines in biomedicine. World Scientific, 2011.

[6] Ana O. Alves, Francisco C. Pereira, Assaf Biderman, and Carlo Ratti. Place enrichment by mining the web. In *Ambient Intelligence*, volume 5859 of *Lecture Notes in Computer Science*, pages 66–77. Springer Berlin Heidelberg, 2009.

[7] Alan Bensky. *Wireless positioning technologies and applications*. Mobile/Wireless Communications. Artech House, 2007.

[8] Christos Bouras and Vassilis Tsogkas. Improving text summarization using noun retrieval techniques. In *Knowledge-Based Intelligent Information and Engineering Systems*, pages 593–600. Springer, 2008.

[9] Allan Brimicombe and Chao Li. *Location-based services and geo-information engineering*. Mastering GIS: Technol, Applications & Mgmnt. Wiley, 2009.

[10] Guiseppe Carenini, Gabriel Murray, and Raymond Ng. *Methods for mining and summarizing text conversations*. Synthesis digital library of engineering and computer science. Morgan & Claypool, 2011.

[11] Ruizhi Chen. *Ubiquitous positioning and mobile location-based services in smart phones*. Premier reference source. Information Science Reference, 2012.

[12] Jongpill Choi, Minkoo Kim, and Vijay V. Raghavan. Adaptive relevance feedback method of extended boolean model using hierarchical custering techniques. *Inf. Process. Manage.*, 42(2):331–349, March 2006.

[13] Heting Chu. *Information representation and retrieval in the digital age*. ASIST monograph series. American Society for Information Science and Technology, 2003.

[14] Julien Eberle and Gian Paolo Perrucci. Energy measurements campaign for positioning methods on state-of-the-art smartphones. In *Consumer Communications and Networking Conference (CCNC), 2011 IEEE*, pages 937–941, Jan 2011.

[15] Mike Gavaghan. Java geodesy library for GPS – Vincenty's formula. http://www.gavaghan.org/blog/free-source-code/geodesy-library-vincentys-formula-java/. Accessed: 2015-01-11.

[16] Daniel Jacob Gillick. *The elements of automatic summarization*. PhD thesis, University of California, Berkeley, 2011.

[17] Ayan Goker and John Davies. *Information retrieval: Searching in the 21st century*. Wiley, 2009.

[18] Martin Hassel. Exploitation of named entities in automatic text summarization. In *Proceedings of NODALIDA*, volume 3, 2003.

[19] Djoerd Hiemstra and Arjen P. de Vries. Relating the new language models of information retrieval to the traditional retrieval models. (TR-CTIT-00-09), 2000.

[20] Djoerd Hiemstra and Arjen P. De Vries. Relating the new language models of information retrieval to the traditional retrieval models. Technical report, Centre for Telematics and Information Technology, 2000.

[21] Jeffrey Hightower and Gaetano Borriello. Location sensing techniques. *IEEE Computer*, 34(8):57–66, 2001.

[22] Arthur V. Hill. *The encyclopedia of operations management: A field manual and glossary of operations management terms and concepts*. FT Press, 2012.

[23] Ryan Hill and Janet Wesson. Using mobile preference-based searching to improve tourism decision support. In *Proceedings of the 2008 Annual Research Conference of the South African Institute of Computer Scientists*, SAICSIT '08, pages 104–113. ACM, 2008.

[24] Stacy F. Hobson. *Text summarization evaluation: correlation human performance on an extrinsic task with automatic intrinsic metrics*. University of Maryland, College Park, 2007.

[25] Ville Honkavirta, Tommi Perala, Simo Ali-Loytty, and Robert Piche. A comparative survey of WLAN location fingerprinting methods. In *Positioning, Navigation and Communication, 2009. WPNC 2009. 6th Workshop on*, pages 243–251, March 2009.

[26] Andreas Hotho, Andreas Nürnberger, and Gerhard Paaß. A brief survey of text mining. *Ldv Forum*, 20(1):19–62, 2005.

[27] Christian Huemer and Pasquale Lops. *E-Commerce and web tchnologies: 14th international conference*. Lecture notes in business information processing. Springer Berlin Heidelberg, 2013.

[28] Diana Inkpen. University lecture: TF-IDF (term frequency/inverse document frequency) weighting example. University of Ottawa. School of Electrical Engineering and Computer Science, 2014.

[29] Frank Ivis. Calculating geographic distance: Concepts and methods. In *NorthEast SAS Users Group Conferences. Data Manipulation*, 2006.

[30] Elliott Kaplan and Christoper Hegarty. *Understanding GPS: Principles and applications, second edition*. Artech House, 2005.

[31] Hassan Karimi. *Telegeoinformatics: Location-based computing and services*. Taylor & Francis, 2004.

[32] Pantea Keikhosrokiani, Norlia Mustaffa, Nasriah Zakaria, and Muhammad Sarwar. Wireless positioning techniques and location-based services: A literature review. In *Multimedia and Ubiquitous Engineering*, volume 240 of *Lecture Notes in Electrical Engineering*, pages 785–797. Springer Netherlands, 2013.

[33] Eamonn Keogh. University lecture: Naive Bayes classifier. University of California - Riverside. Computer Science & Engineering Department, 2014.

[34] Eugene F. Krause. *Taxicab geometry: An adventure in non-euclidean geometry*. Dover Books on Mathematics Series. Dover Publications, 1986.

[35] Ela Kumar. *Natural language processing*. I.K. International Publishing House, 2011.

[36] Axel Küpper. *Location-based services: Fundamentals and operation*. John Wiley & Sons, 2005.

[37] Mathias Lemmens. *Geo-information: Technologies, applications and the environment*. Geotechnologies and the Environment. Springer, 2011.

[38] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, 2004.

[39] Werner Mansfeld. *Satellitenortung und Navigation: Grundlagen, Wirkungsweise und Anwendung globaler Satellitennavigationssysteme*. Informations- und Kommunikationstechnik. Vieweg+Teubner Verlag, 2009.

[40] David Martens and Foster Provost. Explaining documents' classifications. *Center for Digital Economy Research*, 2011.

[41] Manel Martínez-Ramón and Christos Christodoulou. *Support vector machines for antenna array processing and electromagnetics*. Synthesis Lectures on Computational Electromagnetics. Morgan & Claypool Publishers, 2006.

[42] Gattullo Michele, Di Donato Michele, and Sorrentino Fabio. VisitAR: A mobile application for tourism using AR. In *SIGGRAPH Asia 2013 Symposium on Mobile Graphics and Interactive Applications*, SA '13, pages 103:1–103:6. ACM, 2013.

[43] Eleni Miltsakaki and Audrey Troutt. Real-time web text classification and analysis of reading difficulty. In *Proceedings of the Third Workshop on Innovative Use of NLP for Building Educational Applications*, pages 89–97. Association for Computational Linguistics, 2008.

[44] Juan Manuel Torres Moreno. *Automatic text summarization*. ISTE. Wiley, 2014.

[45] Wolfgang Narzt. A generic context-based architecture for energy-efficient localization on mobile devices. In *Proceedings of International Conference on Advances in Mobile Computing*, MoMM '13, pages 33:33–33:42, New York, NY, USA, 2013. ACM.

[46] Ani Nenkova and Kathleen McKeown. Automatic summarization. In *Foundations and Trends in Information Retrieval*, volume 5, pages 103–233, 2011.

[47] Jian-Yun Nie. *Cross-language information retrieval*. Synthesis lectures on human language technologies. Morgan & Claypool Publishers, 2010.

[48] Kamal Nigam. Using maximum entropy for text classification. In *In IJCAI-99 Workshop on Machine Learning for Information Filtering*, pages 61–67, 1999.

[49] Kamal Nigam, John Lafferty, and Andrew McCallum. Using maximum entropy for text classification. In *IJCAI-99 workshop on machine learning for information filtering*, volume 1, pages 61–67, 1999.

[50] United States Department of Homeland Security. GPS constellation status for 12/22/2014. http://www.navcen.uscg.gov/?Do=constellationStatus. Accessed: 2014-12-22.

[51] Margherita Pagani. *Encyclopedia of multimedia technology and networking*. Encyclopedia of Multimedia Technology and Networking. Idea Group Reference, 2005.

[52] Narayan Panigrahi. *Computing in geographic information systems*. CRC Press, 2014.

[53] Chuan-Chin Pu, Chuan-Hsian Pu, and Hoon-Jae Lee. Indoor location tracking using received signal strength indicator. *The Emerging Communications for Wireless Sensor Networks*, pages 978–953, 2011.

[54] Julia J. Quinlan. *Latitude, longitude, and direction*. How to Use Maps. Rosen Publishing Group, 2012.

[55] Grzegorz Sabak. Tests of smartphone localization accuracy using W3C API and Cell-Id. In *Computer Science and Information Systems (FedCSIS), 2013 Federated Conference on*, pages 845–849, Sept 2013.

[56] Catarina Silva and Bernadete Ribeiro. *Inductive inference for large scale text classification: Kernel approaches and techniques*. Studies in Computational Intelligence. Springer, 2009.

[57] Roger W. Sinnott. Virtues of the Haversine. *Sky and Telescope*, 68:158, 1984.

[58] Richard Sproat and Joseph Olive. Text-to-speech synthesis. In *Handbook of Signal Processing*. CRC Press, 1999.

[59] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. YAGO: A core of semantic knowledge unifying WordNet and Wikipedia. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, pages 697–706. ACM, 2007.

[60] Eiichiro Sumita and Fumiaki Sugaya. Word pronunciation disambiguation using the Web. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, NAACL-Short '06, pages 165–168, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.

[61] Emiliano Trevisani and Andrea Vitaletti. Cell-ID location technique, limits and benefits: An experimental study. In *Mobile Computing Systems and Applications, 2004. WMCSA 2004. Sixth IEEE Workshop on*, pages 51–60, Dec 2004.

[62] Thaddeus Vincenty. Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations. *Survey review*, 23(176):88–93, 1975.

[63] Jonas Willaredt. Wi-Fi and Cell-ID based positioning-protocols, standards and solutions. *SNET Project WT*, 2011.

[64] Michael Worboys and Matt Duckham. *GIS: A computing perspective, second edition*. Taylor & Francis, 2004.

[65] Chun-Chieh Wu, Yanpeng P. Cai, and J. Zhao. Adaptive power saving strategy for GPS intelligent terminal. In *Geo-Information Technologies for Natural Disaster Management (GiT4NDM), 2013 Fifth International Conference on*, pages 31–35, Oct 2013.

[66] Fei Wu and Daniel S. Weld. Open information extraction using Wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 118–127. Association for Computational Linguistics, 2010.

[67] Guochang Xu. *GPS: Theory, algorithms and applications*. Springer, 2007.

[68] Vasileios Zeimpekis, Panos Kourouthanassis, and George M. Giaglis. Mobile and wireless positioning technologies. *Telecommunication Systems and Technologies, UNESCO Encyclopaedia of Life Support Systems (EOLSS)*, 6.108, 2007.

[69] Reza Zekavat and Michale Buehrer. *Handbook of position location: Theory, practice and advances*. IEEE Series on Digital & Mobile Communication. Wiley, 2011.

[70] Orli Zuravicky. *Map math: Learning about latitude and longitude using cordinate systems*. PowerMath Series. PowerKids Press, 2005.