Master's Thesis
Word assistant app with speech recognition

Ann-Katrin Hannemann

Date of defence:
26th of April 2018

Supervising Professor: Enric Mayol Sarroca
Departament d'Enginyeria de Serveis i Sistemes d'Informació

Master in innovation and research in informatics
Facultat d'Informàtica de Barcelona (FIB)
Universitat Politècnica de Catalunya (UPC), BarcelonaTech

Contents

1 Abstract

In view of more than 30 years of Erasmus and an increasing multi language work life at least in bigger companies the conversation between people with different culture and native languages is frequent occurrence. Especially during an exchange like Erasmus this kind of conversation is common and connects people all over Europe.

During this exchanges people with different language skills and various cultures meet and talk to each other. Even some people from the same country could be grown up with different background. That's a possibility for a lot of misunderstandings. In addition e.g. sometimes a similar word have different meanings in different languages as well.

The proposal of this thesis is to develop a mobile app to assist this people with speech recognition to use the most adequate word during a conversation or in an article. The app will provide a subset of alternative words. The speech recognition feature helps the users to search the words during the conversation without interrupting it for typing a lot but typing should be possible as well.

2 Introduction

First I'm analysing an existing problem and explain what motivates me to solve this in chapter 2.1 Problem at this section. This problem exists during conversations of people with different skills and background. In addition that's what motivates me for this work. Furthermore I will examine the potential help of an app I'm developing for my master thesis in chapter 2.2 Purpose.

2.1 Problem

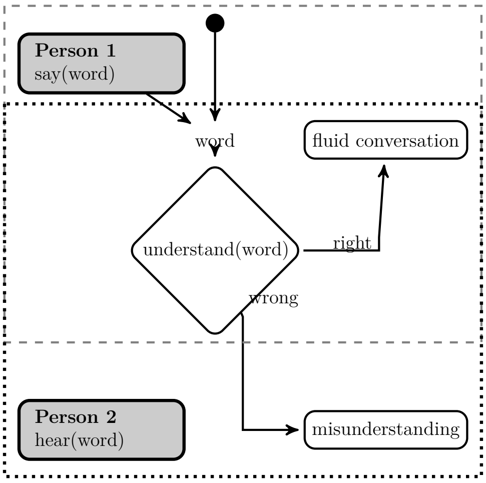


Figure 1: Problem of misunderstanding

In this section I will identify the problem on the basis of several scenarios. A hazard of most of these scenarios explain reasons for misunderstandings. Picture 1 shows the scenario of a conversation. In the good case both persons understand the same and have a fluent conversation, in the other case they have a different context or knowledge. That can lead to a misunderstandings.
In this chapter we regard different aspects in the English language. Firstly there exist some words which are spelled the same or sound similar while meaning something different. Secondly is the opposite, that different words have the same meaning. The people may know the word that their conversational partner uses.
A risk at foreign languages are ”false friends”, words that sounds similar to an other language but have a total different meaning. In addition to this the pronunciation depends on different factors and is able to let words sound like other words that are not meant.
An other significant problem is that nescience in some terms. One person wants to talk about something and doesn't know all necessary words because it's a non-native speaker or has a blackout. Another aspect are technical terms and the limitation of the vocabulary.

Scenario: Homonyms

Homonyms are words that doesn't mean the same even though the same spelling (homographs) or sound (homophones). [92], [93] An easy example for the same word with different meaning this is ”Jaguar”. This is easy to differ during the conversation because of the total different context. On one hand it's an animal. On the other hand it's a car. More difficult is e.g. ”capital”. It can bewilder that in the English language are not only various meanings for this word. It's an adjective and a noun. According to this it could be used as different parts of speech which the conversational parter maybe not expect.

On one hand ”capital” as adjective can stand for the most significant. On the other hand for superior. When a hearing person (Person 2 in picture 1) thinks on another meaning it is harder for them to follow the conversation. The benefit thereby is the probability to notice the misunderstanding earlier. E.g. the writing a character as upper case is called ”capital”. In contrast to this are financials, where it describes a possible investigation of money e.g. in a company. The possible meanings of ”capital” are not over with this examples. It is important to mention that ”capital” is the house where the government meets and like the adjective the official government seat. [92]
An other example for homonyms is ”Turkey”. It's possible a country, an animal or slang for a person (a fool) [70][50]. ”Lead” could be the metal or main role [92]. Instances for homophones are to/two/too, lose/loose and by/buy/bye. [92]

Scenario: Tautology and synonyms

Languages are complex and often there are more words to describe the same thing. This leads to the following problem: During a conversation both dialogue partner know the same thing but using different words. That can make the people feel insecure, so they are not sure if it is the same they are talking about or only something close to the topic. Without finding the right synonyms the understanding is not secured.
By searching a word in a dictionary appear a lot of examples. One of them is the German word ”Linsen”. Am I talking about ”Linsen” in German it depends on the context. Is it technically I mean a lens maybe for photography in my camera. Botanical and during cooking it's a lentil. A third possibility are contact lenses. When a person says ”Linsen help me to see better” the dialogue partner find out by the context which ”Linsen” this person is talking about. But he has to know about the different possibilities.
An other example is ”ice”. Imagine it's a sunny day, you and some friends are drinking lemonade in the garden of one friend. Than the host asks ”Do you want some ice?”. It's really hot and the lemonade is not cold any more, so yes. The host goes inside and doesn't come back with ice cubes for your drink. Instead you are getting ice cream. The other way around is possible as well.
Now you think this problem can be solved by using some more words or add a description to say it twice but be sure that the other understands you better. I call this section tautology because a tautology is a rhectorical device where the same meaning is said twice like in ”a dry desert” or ”a short summary”. Sometimes this device induces higher clarity. But there is no guarantee for this argument. It can be confusing as well when someone told you something and starts saying the same again with other words. [94]
This problems are major significant when you are talking in a foreign language or only being tired. This induces the next problem. There are not only words in one language which sound the same with different meaning. There is the same problem when we look to other languages and compare them with each other. The mix of words of different languages is called false friends.

Scenario: False friend

Famous mistakes, when the spoken language of the conversation isn't the foreign language for at least one person, are false friends. That are words that are written or sound similar in different languages. An example for Spanish speaker is the false friend lentillas (lenses) and lentejas (lentils). The word for success in Spanish is ”exito”, therefore it's possible to wish someone ”a lot of exit with an exam” [33] and confuses other people with this false friend. Other famous false friends are ”qué vergüenza” and ”estoy embarazada”.[69][71] [117] [41] [1]

Scenario: Different pronunciation

The previous scenarios were about learned languages and special words in languages. This scenario is different because it depends on the persons which are talking together. People from different places of the world pronounce things different and maybe they have a strong accent or learned wrong pronunciation. This makes the understanding harder and some words sound like others.

A good example for such misunderstanding of different pronunciation are the stories ”The Italian man who went to America” [148], ”The Italian Man Who Went To Malta” (apendix 11.3) [146], [147] and ”The French Man Who Went To Malta” [145]. In one story the Italian wants to order Coca Cola as drink and some Asian food. He says ”Mario is angry/hungry and want a cock/coke in my hot hole/roll” and gets in trouble because the word coke for the drink sounds too similar to a drug and a masculine genital.

Scenario: Nescience and blackout

In some cases non-native speaker like travellers, workers and students from other countries can't remember some words or don't know them in the right language when they need them. In place of a false friend or long paraphrase they say something that sounds near to what they want to say. The following example happened to a friend from Japan. He was in a restaurant and wanted to help two nuns when they asked about meat inside a dish. He found out that there was chicken inside and translated it with ”polla” (dick) instead of ”pollo”.

This kind of blackout can happen everybody. Sometimes even high educated native speaker have a black out and can't remember spontaneous a word which is necessary or describes an issue.

Scenario: Exact description

Among others people are judged by they vocabulary [2]. Authors, academics, public persons like politicians say intimately what they mean and don't only use simple language. They e.g. find exact words, rely on technical terms and a large vocabulary.

The same hold good for students. On one hand they sound more intelligent with a big vocabulary and sophisticated text [113]. On the other hand some words describe something more exact. But often their vocabulary is more limited. They have to learn not only their subject, they have to learn how to speak about it as well to use the right words. Let me give you an small grammar example: ”fewer” and ”less” are synonyms for each other [29] but there is a difference between them. When the student is talking about countable things he should use ”fewer” for a smaller number of it. Is it a smaller amount and not countable he has to use ”less” [92].

An other example is written homework with a limited vocabulary. It doesn’t sound good and is more boring for the reader when all paragraphs start with the same words, the student repeats . [113]

Conclusion of the scenarios

It looks like there are easy solutions for some of the scenarios: to search in a search engine. But that is annoying to search during the conversation in the phone e.g. with Google if you can’t remember a special word besides you get a lot of results in which you have to find the right result.

An other possibility for some of the described problems is to explain the missing word. But this interrupts the conversation as well and stops it for the search. Not all of the described problems can be solved with long descriptions. If a student writes a text and needs a synonym for an overused word he can’t always ask others. In addition the conversation partner do not always notice that there is a misunderstanding or one person doesn’t want to look stupid by asking or searching in his phone.

2.2 Purpose

Like in chapter chapter 2.1 Problem described is the current state of writing and conversations that persons have the risk of misunderstandings or have to change the topic, if the conversation stops because they ask for the right words. This work is the opportunity to prove that people don’t have to use online search engines or ask other people for help.

The goal of this project is the realisation of a mobile application with which a conversation don’t stop or change the topic because of missing words. The user should only says e.g. synonymous and get the missing word by the app.

The following list shows the benefits for different user of the app.

- learning, improve skills, expand vocabulary
- improve texts
- find a missing word
- against blackout
- make conversation easier
- less misunderstandings

This benefits are for authors, academics, public persons when they write ambitious texts. Students can use it for the same reason but most of the time this texts will be for homework, projects, thesis. In addition they improve their skills and expand their vocabulary for later works. Non-native speaker also learn and remember new words therefore it's good for traveller.
Poets and songwriter can improve their work with the app by finding rhymes, rhetorical devices like tautologies and antonyms. In total everybody gains by using the app during a conversation even if it's only to find a missing word without interruptions for guessing or using a search engine.

3 State of the Art

After analysing the problem and the purpose of the word assistant app is to take care of it with a research about previous work, existing technologies and applications, which already cover parts of the problem.
In this section I am going to analyse different kinds of existing technologies in chapter 3.1 Existing technology. Than I will continue with this task on applications in the market which have more or less similar functions to the word assistant app in chapter 3.2 Voice assistant services and chapter 3.3 Similar apps. Most of the analysed apps are available at the Google Play store, because I have searched for android apps.
It's important to check for special elements of this technologies and apps. Thereby it's possible to find more interesting features for the word assistant app. After this I will consider in chapter 3.4 Conclusion which functions are interesting and useful for the word assistant app.

3.1 Existing technology

In this chapter I consider technologies I want to use during this project. I describe tools that influenced my decisions or alternative technologies as well.

Type of app: native, web or hybrid

It exists a rough classification of apps. I use the separation in three types: native, web or hybrid without taking account of the types in between. The following is a short description of the different types. The Table 2 shows the most important differences.

Native The native apps are specific build for a single platform. The user experience and performance are the best because native apps are developed to support the device and their user interface the best and is personalized. The need of a specific language for the development is a disadvantage when the developer has to learn it. Furthermore every platform has an other language. This is one of the reasons why native apps are more expensive, the developer need special skills and have to develop an own app for every platform. For the development of a native app to be worthwhile, they take care to be used for a long time.
A native app is not automatically up to date. Compared to an web app updates by the user are necessary. [31] [78] [82]

Web Web apps are like web sites. The user calls them via an URL and doesn't download them at an app store. Therefore he execute it in a web browser. This has effect on the user experience, because it's more outlay to start the app, has no access of device features and looks the same on every platform which means there is a difference to other apps on the user's device.
The performance of web apps is slower than the other app types because everything has to be requested by the internet. This is a disadvantage that includes the advantage of being always up to date. [31] [78]

Hybrid Hybrid apps are a combination of web and native apps. They work cross platform like web apps while behaving like native apps. A wrapper enable the platform specific features.

All advantages and disadvantages depend on the degree how similar the hybrid app is compared to the native and web app. E.g. their performance is slower and they are less interactive than native apps because they have to load web content. A hybrid app is more expensive than a standard web app. The more native it is, the more expensive is the hybrid app.

The mix of web and native app for the different platforms makes the development faster than for native apps. The wrapper for native part as bridge between platforms and the webview support everything of HTML5. These mix admittedly makes the bug fixing is more difficult. [31] [78]

	native	**web**	**hybrid**
platform	single	multi (browser)	multi
internet access	not always required (depends on the functionality)	indispensable	necessary, less than web
performance	best	slow	middle
distributed	app store	browser	app store
user experience	great: natural for platform (interactive, intuitive)	poor: typing of URL necessary, unusual for platform	good: depends how native, like native apps possible
device features	yes	no	yes, through wrapper
development	specific languages per plattform	JavaScript, CSS, and HTML5	single code base
costs	expensive	inexpensive	cheaper than a native app
updates	user has to allow it	automatically	partly: web content automatically, stored part user related

Table 1: Comparison of app types [31] [78] [82]

Platform

Mobile platforms are Android, iOS, Windows Phone and BlackBerry OS (works with android since 2015) [32]. 99.6 % of the market are Android and iOS [17] [10]. It is possible to develop apps for multi platforms but in comparison single platforms have different advantages. Often the performance and security are better. The user experience can be better customized.[78]. Therefore I analyse the differences between developing an app for an Android or iOS platform.

	Android	**iOS**
Target devices	Google devices	Apple devices
IDE (examples)	AppInventor, Eclipse and AndroidStudio	XCode, CodeRunner, App-Code, Swifty
OS of development device	any (Windows7) at hand	Mac computer to rent
programming language **Knowledge**	Java already experiences	Swift to learn
Test device (real)	at hand	to rent
Test device (virtual)	in IDE	in IDE

Table 2: Comparison of development for the Andoid and iOS platforms [11] [13] [28] [31] [32] [66] [81] [98] [114]

Android is an operating system for Google devices among others for mobile phones and tablets. Apps are developed in Java with Android specific extensions. For developing are different freeware IDEs possible e.g. AppInventor, Eclipse and AndroidStudio. They are running on multi platforms like Linux, Windows and Mac. It is no special operating system or device needed. Testing without a real device is possible with an Android emulator which is available in the IDEs. [31] [66] [98] [114]

iOS is the operating system for Apple devices. In contrast to android development it has an own programming language named Swift. It is similar to Objective-C. The mentioned IDEs in table 2 support the developer with Swift but of those only XCode is free, but a Mac computer is indispensable to run this IDE. [11] [28] [31] [81]

External APIs and libraries

In the following I present APIs (application programming interfaces) and libraries which seem to be helpful for the word assistant app. Which APIs and libraries actually useful for the word assistant app is described in chapter 3.4 Conclusion.

Android's Speech To Text API Google developed an API for the Android operating system that converts speech to text. The Android user can downlad offline language packages and enable an offline mode at his phone settings to use this without internet connection for all apps which are using this API. [18] [62] [47] [112][88]. This speech recognition is not planned for continuous voice recognition because it can be used on- and offline. In the online mode it streams the recorded audio the whole time of the speech recognition as input to a server. At the server the input will be convert into text and send back to the app. That's why it shouldn't be used steady, because this cost a lot of energy and bandwidth. [54] [30] [18] [88] [99] [112] [47] [25]

Thesaurus (WordNet) As the name thesaurus suggests, this API is a dictionary for synonyms. The Thesaurus web service, also known as WordNet, retrieves lists of synonyms from the thesauri dictionaries of OpenOffice in different languages. For the request of synonyms for one word the language has to be selected and sent together with an API key and the word as parameter in a HTTP GET message. The supported languages are Italian, English, French, German, Spanish and Portuguese. The respond is either in a XML or a JSON format. [100] [136] [142] [156]

Big Huge Thesaurus The basis of the Big Huge Thesaurus data is formed by the WordNet database of the Princeton University. Therefore the Big Huge Thesaurus API returns a kind of an extended version of this for the English language. The upgrade are data from an open-source machine-readable pronunciation dictionary for North American English [84] and entries of an internet community [151] [152]. Like WordNet the Big Huge Thesaurus API requires an API key. But the language is limited to English. Another difference is that the Big Huge Thesaurus outputs a list of antonyms, rhymes and parts of speech in addition to the list of synonyms. The server returns this as XML, JSON, PHP or plain text. [121] [152]

Wordnik API Wordnik is an English dictionary website. This dictionary contains definitions, synonyms, antonyms, uses in sentences and pronunciations [21]. The dictionary is continuous growing with additional dictionaries and features [121]. Knicker is a Java library for the access of the Wordnik API. It is written by Jeremy Brooks and only needs an API key for the usage. [20] [121]

TextRazor The TextRazor API analyse the content of a website to extract the meaning of the text there. The result is a classification of the text with categories, topics and entities. The context is shown by analysing synonyms, relations and typed dependencies. Furthermore it is possible to tag links, like one to the wikipedia page, to give more information about entities such as custom products, popular people and companies. [138] [139]

The API provides Client SDKs in Python, Java, and PHP. [140]. TextRazor supports different languages, amongst others English, Spanish and German. But a short text may not enough context to detect the right language automatically. [137]

JWKTL (Java-based Wiktionary Library) Wiktionary is a free web-based dictionary. It has its seeds in the idea of a lexical companion to Wikipedia. Among others it contains synonyms, antonyms, phrases, sense definitions, etymology, example sentences, translations and semantic relations.

For the usage lists of words in different languages can be downloaded as XML to include it in a database. They are continually growing and constructed by volunteers in a collective. Some of this lists are available in a SQL script form as well. [161] [162] [160] [155] [24] [128]

3.2 Voice assistant services

Voice assistant services are intelligent assistants which the user controls with his voice. There are many different voice assistant services. During the research I focused on five popular services by big companies: Siri by the Apple Inc. ([15] [16] [14] [12]), Amazon Echo (a.k.a. Alexa) by the Amazon Inc. ([8] [7] [6]), Cortana by Microsoft ([102] [101] [105] [103] [106]), the Google Assistant ([56] [59] [57] [58] [143] [40] [39] [97] [55]) and Samsungs Bixby ([124] [122] [123] [125] [126]). A description with more details is in the appendix at page 100.

Most of them are not only controllable by voice. Tapping and texting are working as control as well. Some voice assistant services like Siri and Bixby have an user interface and for the others exists an app for the smartphone. There it is possible to correct the spoken input with a tap and typing in the case that the understanding was wrong.

The use of the voice assistant service only with the voice without anything else makes a faster and hand-free usage accessible which is one of the main reasons to use voice assistant services. To support this they have an alternative way to start the services only by saying a code word and the voice assistant services are always listening for this. An user can start the voice control with a start sentence like ”Hey Siri”, ”Alexa”, ”Ok Google”, ”Hey Cortana” and than the spoken request. Thereby all analysed voice assistant services understand at least the English spoken language.

It looks like the basic functions, that all voice services have, are the calendar with a reminder, music player, weather and news report, calling and messaging. To answer questions of the users they use different search engines for example Google (Google Assistant) and Bing (Amazon Echo) [35]. For example the Google Assistant is a kind of combination of the Google search engine and Google Now to fast up the support of it’s user and give him individual results more quickly. It adapt itself to the interests of it’s user.

This is more special in comparison to other applications and in addition all the voice services are learning continuous to get smarter and support their user. Except of studying the behaviours and routines of them the voice assistant services combines informations like the time, the location and context knowledge between different apps to be good assistants. E.g. most voice services may remind their user when he comes to the office to call someone back or to buy milk when he is next to the supermarket. Depending on the location, time and the traffic conditions the assistant tells his user when he has to leave for upcoming appointments as well.

Nevertheless the services are not only connecting informations from a handful apps. The functions of some voice assistant services are expandable by more applications or devices of other supplier and devices. The compared voice assistant services are all compatible with different devices and can synchronize between applications. That makes them good for e.g. supporting a smart home. Indeed some of the voice assistant service devices are made to stay at home and can’t be taken everywhere e.g. Amazon Echo and Google Home.

3.3 Similar apps

There are different kinds of apps which are similar to the word assistant app but only cover some required functions. This thesis is handling android apps, therefore most of the apps are provided by the Google Play Store [49] and have a high ranking of other users there.
In the following the most important apps will be described briefly. First I decide which kind of app could be interested. After this I will describe the main functions of selected applications. After this I will consider in chapter 3.4 Conclusion which functions are interesting and useful for the word assistant app.

Rhymes I consider the app B-Rhymes [90][89]. Alternative apps for rhyming dictionaries are e.g. "RhymeZone Rhyming Dictionary" and "Rhyme Reverse Dictionary". The usage is the search of rhymes for English words with similar pronunciation e.g. for songs and poems. Because they help during the hunt for antonyms and homophone (key/quay) which sound the same by different meaning.

Translations To find a missing word in an other language or expand the vocabulary dictionaries which find translations nearly in realtime support the conversations or help during texting. Therefor I consider the following apps: Yandex.Translate [159], Google Translator [50], Microsoft Translator [104], Reverso Context[131] [132].

Synonymous Not only a translation helps to make a conversation easier. Synonyms and antonyms help to decrease misunderstandings. Apps to look for synonyms and antonyms are: offline Thesaurus [43], Dictionary Synonyms & Antonyms [118], Synonym & Antonym Dictionary [96] and Synonyme (Alternative) [45]

Idioms & Phrases Depending on society and culture every language has other sayings, common phrases, idioms, proverbs and idiomatic expressions which are used in everyday conversations [107]. To speak properly and improve the language skills it helps to know and use them. Misunderstandings in spite of a good vocabulary are possible when a person doesn't know this common phrases. Therefore I analyse two apps: English Idioms & Phrases [107] and All English Idioms & Phrases [108].

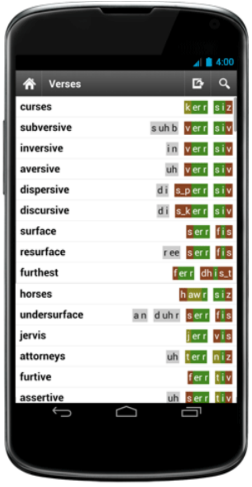
Spell or Grammar Check Correct grammar and spelling without mistakes are helpful for a plain conversation. Most of the time your conversation partner understand you even when you do mistakes. Therefore this kind of checking apps are only considered if they are included in the other apps with functions like translation.

Games By playing games you can expand your vocabulary and improve your skills. I only found games where the user guess the words and not the app. This apps are e.g. TABU/No Lo Digas [9] and GuessUp Party Charades [26]. This games are different to the idea of this work and will not be described in the following text.

B-Rhymes

Kind of the app	Rhyming dictionary
Offline usage	No difference by searching
Input	Random search, text input, selection of result word.
During the input	Word suggestion
Choosing a word	Selection at suggestion list or typing a whole word. Selection of previous result.
Results	What does and doesn't rhyme, words that sound good together.
Stored data	Search history
More information	Degree of similarity of the pronunciation sound.

Table 3: App B-Rhymes



curses			k err	s i z
subversive		s uh b	v err	s i v
inversive		i n	v err	s i v
aversive		uh	v err	s i v
dispersive		d i	s_p err	s i v
discursive		d i	s_k err	s i v
surface			s err	f i s
resurface		r ee	s err	f i s
furthest			f err	dh i s_t
horses			h aw r	s i z
undersurface	a n	d uh r	s err	f i s
jervis			j err	v i s
attorneys		uh	t err	n i z
furtive			f err	t i v
assertive		uh	s err	t i v

Figure 2: Screenshots of the app B-Rhymes [89].

B-Rhymes is a dictionary for rhymes. It shows more or less rhyming words in a list with scores about the similar pronunciation. That means not all found rhymes are technically rhyming, some only sound good together. The degree of similarity of the rhymes is pictured by colours like traffic lights at each rhyme, figure 2 shows this. [89] [90]

Microsoft Translator

Version	3.1.252
Google Play ranking	4,6 of 5
Kind of the app	Translator
Offline usage	English (default) and other downloaded language packs.
Input	Typing, voice/speech (voice and conversation), camera picture, screenshot
During the input	Realtime translation with suggestion
Choosing a word	Whole input necessary.
Results	Direct translation of sentences. More information by searching only single words.
Stored data	Favourites, search history
More information	Alternative translations and meanings, pronunciation guide. Option to share with other apps.
Size on phone	569 MB with the German and Spanish language pack

Table 4: App Microsoft Translator

The Microsoft Translator app translate the input in more than 60 languages and works for written text, spoken text, conversations and images. The languages are offline available after downloading. The language pack for English is installed per default. Each additional language pack needs between 119 MB and 240 MB (e.g. 234 MB Spanish and 231 MB German, but only 122 MB for Afrikaans, 119 MB for Filipino).
After starting the app the user has to select a translation mode at the start screen. He can choose between a phrasebook, search history and the different kinds of input like voice, text, image or conversation mode with more than one translation direction. The conversation mode makes it possible to translate a bidirectional dialogue. But it is always necessary to press a button who is speaking and which language has to be translated. According to the information on the website the conversation mode can be used by up to 100 people when they add more devices.
With the phrasebook users are able to learn or improve their language skills. The app offers different topics with important phrases e.g. for travelling the Microsoft Translator app is possible as a language guide. Other aids in learning languages are the pronunciation instructions and transliterations of the Microsoft Translator. This is supported with explanations as well while the app provides alternative translations and meanings for single words. [104]

LEO dictionary

Version	7.1.4
Google Play ranking	4,3 of 5
Kind of the app	Translation (dictionary)
Offline usage	No
Input	Voice or text input
During the input	Word suggestion (bilingual)
Choosing a word	Input of the beginning of a word, selection from suggestion of possible words
Results	Bilingual list of translations, sorted by word classes, examples and phrases
Stored data	Search history
More information	Inflection tables (e.g., plural, tenses, etc.), pronunciation, grammar, etymology, explanations, orthographical words
Size on phone	32 MB

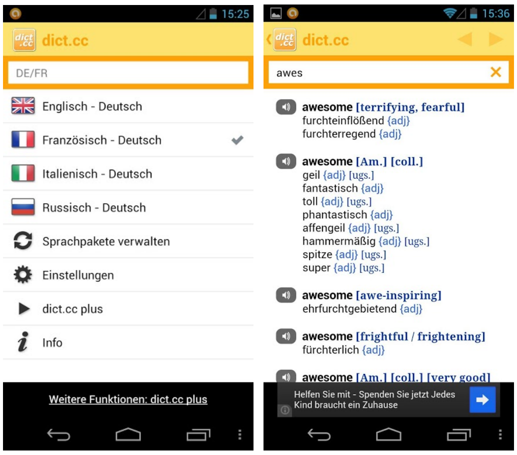
Table 5: App LEO dictionary

LEO is an app from Germany which you can use like an online dictionary for 8 languages. It's the app for the in Germany famous website leo.org [85] and more than 5.000.000 persons installed it on their mobile device.
Most of other translation apps only translate single words. LEO has inflection tables (tenses, basic form, plural), which help with grammar and give informations about etymology. Special on this app is the community which helps with any question about e.g. grammar or translations in a forum. There are explanations as well. [86]

dict.cc

Version	7.0 (free version)
Google Play ranking	4,4 of 5
Kind of the app	Translation (dictionary)
Offline usage	Saved language packs, no voice output
Input	Typing, insertion of copied text
During the input	Word suggestions
Choosing a word	Input of the beginning of a word, selection from suggestion of possible words
Results	List of different possible translations sorted by word classes
Stored data	Saved languages, search history
More information	Re-translation, inflections, word class
Size on phone	With 2 (of the bigger) language packs 167 MB

Table 6: App dict.cc



awesome [terrifying, fearful]
furchteinflößend {adj}
furchterregend {adj}

awesome [Am.] [coll.]
geil {adj} [ugs.]
fantastisch {adj}
toll {adj} [ugs.]
phantastisch {adj}
affengeil {adj} [ugs.]
hammermäßig {adj} [ugs.]
spitze {adj} [ugs.]
super {adj} [ugs.]

awesome [awe-inspiring]
ehrfurchtgebietend {adj}

awesome [frightful / frightening]
fürchterlich {adj}

awesome [Am.] [coll.] [very good]

Helfen Sie mit - Spenden Sie jetzt Jedes Kind braucht ein Zuhause

Figure 3: Screenshots of the app dict.cc [70].

dict.cc is a German translation app with more than 1.000.000 installations and a good ranking at the Google Play Store. The user can choose between 51 language pairs that he has to download for offline usage. The size of the dict.cc app on a phone depends on the installed language packs and further informations like inflections for the different languages.

Figure 3 shows the start screen of the app on the left side. The user can change the language pair for the translation here or download further language packs.

All of these packs are bidirectional and the search is usable with both languages of the pair at the same moment. A change in the settings is not necessary for this.

The view of the results after the input of halve a word is figure 3. When the user is misspelling the word the app offers a correction. A voice output is only online possible. The user interface has many settings for the display and is easy to understand with two different views and common icons. The readability and usability are especially good because of a zoom option, audio and a changeable view between tabular and list view. [70]

Google Translator

Version	v5.15.0
Google Play ranking	4,4 of 5
Kind of the app	Translator
Offline usage	Text translation for downloaded languages
Input	Text and voice input, paste copied text, camera images, handwriting
During the input	Direct translation of input, suggestion of possible correction by spelling mistakes or similar words
Choosing a word	Always immediate translation. When only one word: word suggestions for this.
Results	Direct translation
Stored data	Search history and favourites. Languages of offline translation, Vocabulary: Mark and save translations in any language for future use
More information	Alternate translations.
Size on phone	134 MB

Table 7: App Google Translator

The Google Transltor is an app with many different kinds of input. The online translation for text and voice input works for 103 languages and 59 offline after downloading each of them.

The app has the same features like Microsoft Translator e.g. the conversation mode and image translation. But it has a correction of spelling mistakes or similar words. The Google Translator app is the only app which translate handwriting as well. [50]

Yandex.Translate

Version	3.22 Build 2108
Google Play ranking	4,4 of 5
Kind of the app	Translator
Offline usage	Online translation between any pair of the 90 languages, visual text recognition for 12 languages. For offline usage you can download 7 languages.
Input	Speak words or phrases, take or choose a photo, other apps (Android 6.0 or higher).
During the input	Word suggestion and automated language detection.
Choosing a word	Direct translation of possible searches, choosing of word suggestions possible.
Results	Voice output, translation of whole sentences. When only a single word: informations like word class and transliterations.
Stored data	Save translations in Favourites and view your translation history at any time. Marked words for learning.
More information	Learn new words and their meanings through usage examples in the apps dictionary.
Size on phone	47 MB

Table 8: App Yandex

Yandex.Translate is a translation app from a Russian company. The Yandex app enables the translation of input (text, voice, photos), websites and other apps. Online the app can translate between 80 languages. Without internet connection are only 6 language pairs from/to Russian available. The voice input works for 4 languages and the photo input only online. Yandex.Translate has a language auto-recognition and suggests the used language.

Furthermore the app has a learning part with a kind of virtual vocabulary cards. The user can mark some of his translations and repeat it in this part to memorize them. But the app only shows the vocabulary in one direction with the searching word and not the result. The user can't check whether his memorized word is right or not.

Furthermore the app place more services at their users disposal. The user can get informations about the weather and traffic on the notifications panel and lock screen. This panel includes a search bar which works with voice and text input and opens the browser with an own search engine. [159]

Reverso Context

Version	v7.9.1
Google Play ranking	4,6 of 5
Kind of the app	Translator, learning
Offline usage	Vocabulary and examples
Input	Voice or text input, click previous result
During the input	Suggestions during typing
Choosing a word	Selection of suggestions or without search only from a list of words.
Results	Synonyms, sentences with context
Stored data	Search history
More information	Word class, context with example sentence, definitions, synonyms, conjugation, irregular verbs, pronunciation.
Size on phone	55 MB

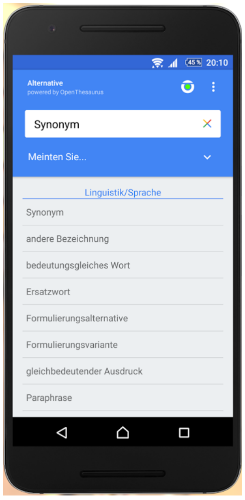
Table 9: App Reverso Translation Dictionary

Reverso context is an other translation app. This app is more about improving the language in an educational way. The user can save translations as favourites in his own phrasebook. To learn the grammar the users use the conjugation tables of the app and searched words from the search history also appear in the learning part of the app. For his vocabularies the user always see expressions and the context.
At this learning part of reverso context the user gets example phases and queries of learned words by doing a quiz and practising the words with flashcards. Even by not using the app directly for learning, the user always get example sentences for his translations and how to use it in context. [131] [132]

Alternative

Version	v1.4
Google Play ranking	4,7 of 5
Kind of the app	Synonym finder
Offline usage	No
Input	Voice or text input, click previous result
During the input	No results shown during the input, The user has to type the whole word
Choosing a word	Input of the whole word and Suggestion of similarly written words.
Results	The synonyms are classified in groups, but often without a caption for any group.
Stored data	Search history, favourites
More information	A wiktionary is integrated
Size on phone	3.3 MB

Table 10: App Alternative



Alternative
powered by OpenThesaurus

Synonym

Meinten Sie...

Linguistik/Sprache

- Synonym
- andere Bezeichnung
- bedeutungsgleiches Wort
- Ersatzwort
- Formulierungsalternative
- Formulierungsvariante
- gleichbedeutender Ausdruck
- Paraphrase

Figure 4: Screenshot of the app Alternative [45]

The app Alternative is powered by Open Thesaurus for synonyms and associations. The app only searches synonyms for words in German but it has a wiktionary which is working with other languages as well.
By using this app the user always starts with the input of a word by texting or voice record (speech). If it's not the first time of using the app there is another option to search a synonym. The user can choose a word which he already searched the time before or if he saved a word as favourite one of this words at a menu. For this word, however he started the search, he can check a proposal of similar written words like ”(jemanden) jagen”, ”Jauer” and ”ja und?” for ”Jaguar”. There is no suggestion of words during the input. Under the button for the proposal is a list to select words, synonyms and compositions of this word. For the word ”Jaguar” there is the first entry for the animal. The word is marked as zoological word by a header which shows the category ”Zoologie”. The second entry is a list of compositions with ”Jaguar”. In this example it's ”Jaguar Cars Ltd” and ”Jaguar”.
Even if no synonym is found but the requested word, the app shows more information of this word in the wiktionary. Finding a synonym is only possible after the insertion of a whole word or selection of one of the proposals. [45]

Figure 4 shows a screen-shot of the main window of the app Alternative. The design is oriented by minimalism. There is nothing useless. The elements are clearly arranged and have good contrasts to each other. The user needs no high effort to understand and use the app. But the Button to open the wiktionary is hard to find intuitively because of the icon. In addition the grouping of the results is not always directly clear because a caption is often missing. [45]
The understandability of the app is normal. A disadvantage of the user interface is that the Button for the wiktionary is not intuitive. Besides this the grouping of the results is not always directly clear.

Offline Thesaurus - Synonyme

Version	5.0
Google Play ranking	4,2 of 5
Kind of the app	Synonyme finder
Offline usage	Contains a database for fast search and auto completion.
Input	Typing in a text field.
During the input	Word suggestion for auto complete.
Choosing a word	Selection in suggested words, results or pressing enter after input of a whole word.
Results	Classified with a caption when differed possible and all words are ordered alphabetical.
Stored data	No
More information	Style level like ”colloquial”, ”technical terminology”
Size on phone	54 MB

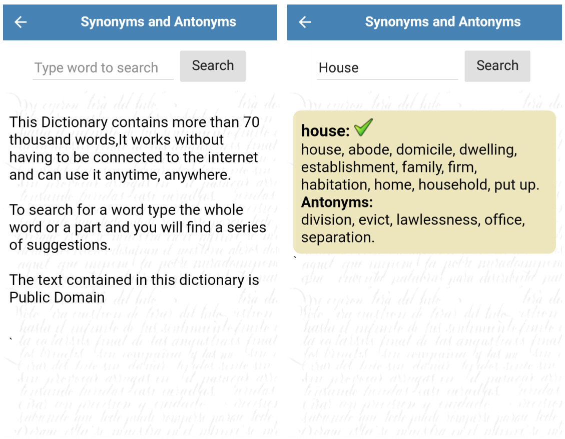
Table 11: App Thesaurus

Offline Thesaurus is an app to find German synonyms. It includes a database for Android with the whole project openthesaurus.de and works without internet connection. The search for synonyms is supported by an auto complete function but doesn’t work when the user is not selecting one of the suggested words, even when there is only one, and only searches the given input from the search text field. Thereby the user interface is very minimalistic. It only has one text field with word suggestion and a list of results below it. [43]

Dictionary Synonyms & Antonyms

Version	1.0.0
Google Play ranking	4,5 of 5
Kind of the app	Synonym and antonym finder
Offline usage	Fully functional
Input	Typing text in text field.
During the input	No word suggestion, search is not starting
Choosing a word	Input of a whole word and without spelling mistakes.
Results	Alphabetically order first synonyms, second antonyms
Stored data	More than 70.000 words
More information	No
Size on phone	57 MB

Table 12: App Dictionary Synonyms & Antonyms



This Dictionary contains more than 70 thousand words,It works without having to be connected to the internet and can use it anytime, anywhere.

To search for a word type the whole word or a part and you will find a series of suggestions.

The text contained in this dictionary is Public Domain

house: ✓
house, abode, domicile, dwelling, establishment, family, firm, habitation, home, household, put up.
Antonyms:
division, evict, lawlessness, office, separation.

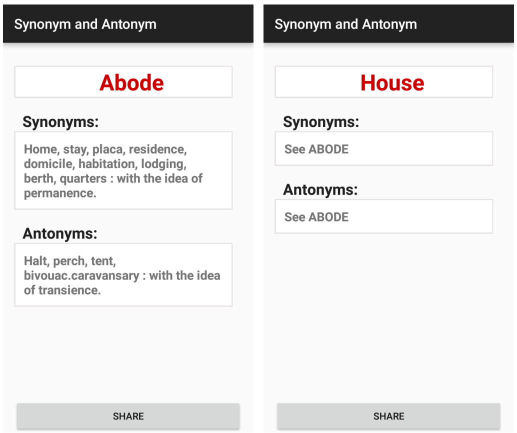
Figure 5: Screenshots of the app Synonym & Antonym Dictionary.

The ”Dictionary Synonyms & Antonyms” app helps the user to find synonyms and antonyms. The GUI is very minimalistic and only consists of the one view with one part that changes when the app shows results. This makes the search easy, but the reset mechanism is confusing.
The figure 5 shows both of this. The first screenshot shows a text field and button to start the search. This is at the second screenshot as well. After a successful search a virtual card with a list of synonyms and one of antonyms appear next to the text field.
The search is without word suggestion. The user always has to write the whole word which he is looking for. In addition the search only works without spelling mistakes and the user can’t see a list of all words of the database. [118]

Synonym & Antonym Dictionary

Version	1.2
Google Play ranking	4,1 of 5
Kind of the app	Synonym and antonym finder
Offline usage	Fully functional
Input	Typing text
During the input	Reducing list of possible suggestions
Choosing a word	Input or from list
Results	2 lists (synonyms and antonyms)
Stored data	50.000 words
More information	No
Size on phone	8.6 MB

Table 13: App Synonym & Antonym Dictionary



Synonym and Antonym

Abode

Synonyms:

Home, stay, placa, residence, domicile, habitation, lodging, berth, quarters : with the idea of permanence.

Antonyms:

Halt, perch, tent, bivouac.caravansary : with the idea of transience.

SHARE

Synonym and Antonym

House

Synonyms:

See ABODE

Antonyms:

See ABODE

SHARE

Figure 6: Screenshots of the app Synonym & Antonym Dictionary.

On the first look this app for finding synonyms is similar to the ”Dictionary Synonyms & Antonyms” app on page 25. Both have a completely database that is offline available and searches synonyms and antonyms for a word. The big difference is the GUI. The ”Synonym & Antonym Dictionary” app on this page offers the user all possible words and has a kind word suggestion, all unfitting words will be removed by typing the searched word. The figure 6 shows the results of the selected word: ”Abode” on the left side.

At the results view is a share button that copies the header (searched word) and the lists of synonyms and antonyms. The view has no back button, the user has to use the one of the smartphone.

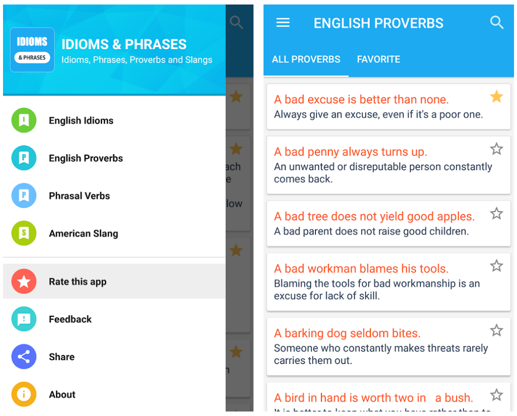
The results can be copied but they are not selectable by the user. This is a problem by the second example which is shown on the right side of figure 6. Instead of showing the synonyms and antonyms of ”House”, the app says ”See ABODE”. To see synonyms that match the user has to remember the other word to search, return to the start page of the app and start a new search. Thereby an other problem occurs, there is no up button and the user has to use the back button of his device.

By comparing the pictures of figure 6 some will notice that ”House” is not a result at the synonyms of ”Abode”. By searching one of the given synonyms or antonyms the previous searched word often is not shown in the results. [96]

English Idioms & Phrases

Version	Version 1.1
Google Play ranking	4,7 of 5
Kind of the app	Learning idioms and phrases
Offline usage	Fully functional
Input	Selection of given idioms/phrases or typing
During the input	Refreshing results
Choosing a word	Direct display (choosing shows more information if possible)
Results	Categories and alphabetically order
Stored data	Favourites
More information	Idioms, proverbs, phrasal verbs, American slang, examples, etymology
Size on phone	29 MB

Table 14: App English Idioms & Phrases



A bad excuse is better than none.
Always give an excuse, even if it's a poor one.

A bad penny always turns up.
An unwanted or disreputable person constantly comes back.

A bad tree does not yield good apples.
A bad parent does not raise good children.

A bad workman blames his tools.
Blaming the tools for bad workmanship is an excuse for lack of skill.

A barking dog seldom bites.
Someone who constantly makes threats rarely carries them out.

A bird in hand is worth two in a bush.

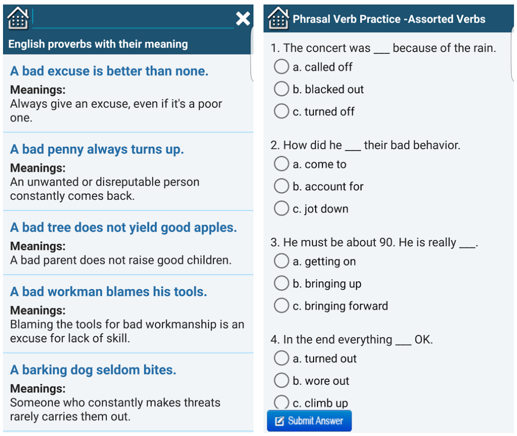
Figure 7: Screenshots of the app English Idioms & Phrases [107].

With the ”English Idioms & Phrases” app users can learn English idioms, phrasal verbs, proverbs among others. He gets explanations, examples and meanings as well.
The main navigation between the functions of the app is by menu. The Figure 7 shows the menu for the navigation on the left side. On the right side is the list of proverbs to see. The first item is marked as favourite and has a filled star as sign. Under every proverb is the meaning given. After clicking on the magnifying glass in the upper right corner the user can start a search after specific entries. [107]

All English Idioms & Phrases

Version	2.0
Google Play ranking	4,5 of 5
Kind of the app	Learning with quizzes
Offline usage	Fully functional
Input	Selection of given idioms/phrases or typing
During the input	Refreshing results
Choosing a word	Direct display (more information by selection)
Results	Categories and alphabetically order
Stored data	Favourites
More information	Idioms, proverbs, phrasal verbs, American slang, meanings and explanations, examples
Size on phone	29 MB

Table 15: App All English Idioms & Phrases



English proverbs with their meaning

A bad excuse is better than none.
Meanings:
Always give an excuse, even if it's a poor one.

A bad penny always turns up.
Meanings:
An unwanted or disreputable person constantly comes back.

A bad tree does not yield good apples.
Meanings:
A bad parent does not raise good children.

A bad workman blames his tools.
Meanings:
Blaming the tools for bad workmanship is an excuse for lack of skill.

A barking dog seldom bites.
Meanings:
Someone who constantly makes threats rarely carries them out.

Phrasal Verb Practice -Assorted Verbs

1. The concert was __ because of the rain.
○ a. called off
○ b. blacked out
○ c. turned off

2. How did he __ their bad behavior.
○ a. come to
○ b. account for
○ c. jot down

3. He must be about 90. He is really __.
○ a. getting on
○ b. bringing up
○ c. bringing forward

4. In the end everything __ OK.
○ a. turned out
○ b. wore out
○ c. climb up

Figure 8: Screenshots of the app All English Idioms & Phrases [108].

Like with the app "English Idioms & Phrases" on page 27 "All English Idioms & Phrases" support his users by learning English. An additional feature are the idioms and phrasal verb quizzes. [108]

3.4 Conclusion

After analysing technologies, voice services and existing apps I want to summarise the conclusions of the state of art respect of the problems and scenarios I identified.
Some stakeholder are still using sources that are made of paper. I didn't explain these in this chapter, as I assume that the respective sources such as dictionaries and lexicons are well known. They also only cover a specific area and are more time-consuming to search.
At the analysis of the technologies I decided to use the following libraries:

Big Huge Thesaurus which bases of the Thesaurus (WordNet) database. For later when other languages are supported the original WordNet database has to be requested. But for English this delivers more results.
Wordnik API as English dictionary by Knicker.
TextRazor already supports different languages, but first I only use English.

The Voice assistant services have the goal to make the usage of their features easier and faster. They support a hands-free usage with the control by voice. To do this at the word assistant app I will use the Android's Speech To Text API.
In the case that the understanding was wrong sometimes it is necessary to type the input. Some of the voice assistant services have an user interface where the user correct his input with tap and typing. Otherwise he can repeat he whole input by voice.
The observed voice assistant services are always listening and have words to start this voice control e.g. with "Hey Siri" and than the spoken request. To save energy the app should only listen when necessary. The word assistant app should start recording after pressing a button.
By comparison of different apps most user interfaces were minimalist, clearly arranged, had a good contrast and for the usage was no high effort necessary. At some apps, e.g. the Microsoft Translator the user has to select a type of input first before he can use the other functions. This is different to other apps. E.g. the user of dict.cc has to select the language pair for the translations, before he starts or uses the selection of the last time. The user gets fast results when his first action is the input and the app works with default settings. Therefore it is helpful when there are not so many possibilities.
To assistant the user most apps have a word suggestion for the input. A few of them already change the output belonging to this during the input. Thereby some apps don't need an user-defined input, the user may select from a list of all entries. The word suggestion during the input is sometimes better to use than in other apps. Some apps only support a limited list of input and don't handle spelling mistakes. The apps which suggest variations of spelling are more helpful.
At many apps the user can choose the result as input for the next search or he gets more information by taping on a result. Some of the apps have example sentences for a word, especially the apps to learn phrases.
The word assistant app should be one app for more features but without to much informations. Some apps have functions that are hidden or the user always needs to use the menu and do some clicks to get his information. That should be improved at the word assistant app.
The translation apps are already well represented in the marked. Especially the Microsoft Translator and Google Translator with a lot of different types of input. Among others they already have a conversation mode where users talk to each other in different languages and get the translation of their conversation partner in their own language. A disadvantage at some of the translation apps is that they only deliver alternative words at the input of single words. After this analysis the focus of this work is more at finding synonyms, antonyms and the context to support the understanding without translating.

4 Proposal and definition of the project

First I define the scope of the project in chapter 4.1 Scope. After this I decide my proposed goals in chapter 4.2 Objectives. This includes technical, functional and business objectives.

4.1 Scope

In this chapter I describe the idea and functionalities of this project. It deals with the specification, design, implementation and evaluation of a mobile application. The app is written for an android operating system with speech recognition. In general the idea of an app with spoken input is created because of the purpose against the problem of interrupted conversations.

Besides the app works with the input of whole sentences instead of only one word. As well it's not a whole translation and the result will be one word that the user can select. Therefore the first part of the project is to decide which words of whole sentences are the most important.

With this words the search of e.g. synonyms and translations can start. That's the second part of the project. After this and finally the results should be shown in a reasonable order by a readable design.

This app will not be a typing corrector nor an spell-checker. In addition the app is no part of a previous software, but different existing libraries will be used during the development of the new android app.

4.2 Objectives

An objective is the description of the goals which should be aimed. In this chapter I describe the goals of the project.

4.2.1 Main objective

- The main objective of this project is the conceptual design and realisation of an innovative mobile app which finds special words and supports conversations and writing.

4.2.2 Business objectives

[134] [72]

- The app should not be published in an app store. The goal is not to earn money or publicity. It's collecting operating experience and knowledge for later projects.
- An other goal is productivity: the app should be helpful for the different user groups which are described in chapter 2.2 Purpose.

4.2.3 Technical objectives

- The first technical objective is to plan a whole software project by my own. This has to be finished in a given time and I need to calculate it with enough time for risks and possible problems.
- Other technical objectives are specifying, designing and implementing the front-end (UI) and back-end of the android app. To satisfy this it's important to define and manage the requirements of a software system.
- An important technical objective is to learn how to implement an useful android app.

4.2.4 Functional objectives

- Like in chapter 2.2 Purpose described the problem with other solutions is the the interruption of the conversation. Therefore a functional objective is to decrease the interruption of the conversation by using speech recognition.
- Furthermore a goal is to create an android app with good user experience.
- The better the algorithm to decide more important words of a sentence is, the better the app. That is why the develop of this algorithm is a functional objective.
- The same reason applies for the order of the results by their importance.

5 Project management

This chapter describes the planning of this project. Look at the whole product life cycle for the word assistant app, this is important to mention the environmental, economic and social impacts.
Afterwards follows the arrangement of a timetable in chapter 5.1 Schedule with tasks belong to the timetable and how long each task needs. Later in chapter 9 Verification and validation this schedule will be compared with the reality.
The next fact to plan after the required time is the budget. This is described in chapter 5.2 Costs. The used methodology for this project to work with this schedule and budget is described in chapter 5.3 Methodology.

5.1 Schedule

The project duration is 26 weeks. The project starts at the 27th of September with the registration of the project. From there it ends at March 2018. The project is implemented by one person which assume different roles. In this case there are six roles: project manager, requirements engineer, systems architect, designer, software developer, software tester.

Role	**Tasks**	**Working time in %**
Project manager	Analyse Requirements Design, implementation and test Validation	25 %
Requirements engineer	Analyse Requirements	15 %
Systems architect	Requirements	10 %
Designer	Requirements Design, implementation and test	10 %
Software developer	Design, implementation and test	30 %
Software tester	Design, implementation and test Validation	10 %

Table 16: Project time of the roles for the different tasks.

The *project manager* is responsible for the project. The *requirements engineer* define, document and maintain the requirements for the word assistant app. The *systems architect* defines the structure of the application with this requirements and the *designer* think about the application should look and the best way of usability. An other role of this person is the development as *software developer*, an other name for this role is programmer. The software developer writes the code and implement the given requirements. The last role is the *software tester*. He review the application in comparison to the requirements.

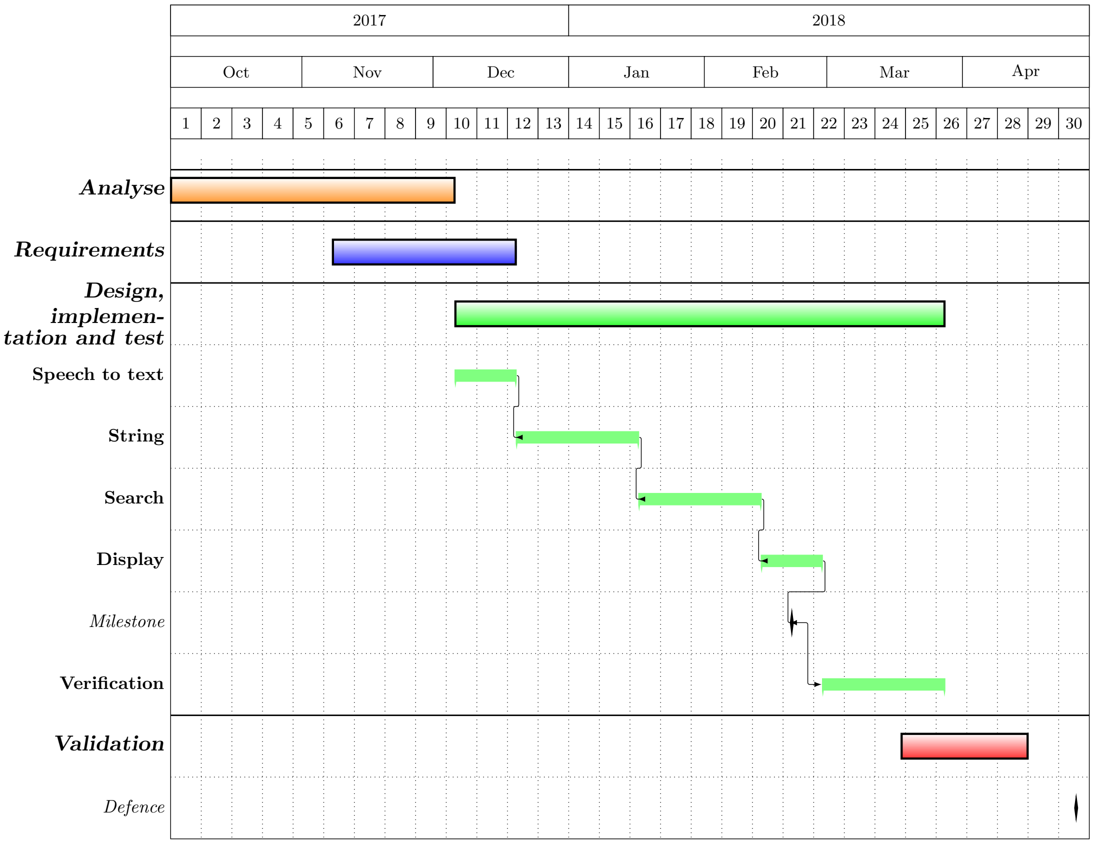


Figure 9: Gantt chart timetable for thesis

The table 16 shows the tasks that are planned for this project. The tasks above are assigned to the different roles and the table shows the percentage of their working time. The figure 9 pictures in a Gantt chart. This chart shows the schedule when the tasks of the table 16 are planned. First the project start with an analysis what applications and technologies already exist. With this knowledge the requirements engineer decide the functions for the word assistant app. This tasks overlap because the requirements engineer compare his plans with the state of the art and decide what makes the word assistant app better and unique. For all intents and purposes this task takes time until the end of the project because the requirements can change by more knowledge, after user review and belong to the future aspects for the thesis. The figure 9 only shows the main task of collecting requirements.
After the analysis the systems architect and designer start with their work. The systems architect takes part in the requirements task to decide together with the role of the requirements engineer what is possible and worthwhile for this project. The designer and software developer work hand in hand. The designer and systems architect define the application, the software developer confer with them and will implement the requirements in a loop of implementing the design and functions. After this the software tester starts his work in this loop. If the software tester finds errors, unexpected behaviours or a

conflict with the requirements he tells this the software developer who has to fix this and give it again to the software tester.

This loop of design, implementation and test runs for all functions. In the schedule they are summarised as ”Speech to text”, ”Strings”, ”Search”, ”Display” and ”Input”. The meaning of ”Speech to text” should be clear. ”Strings” stands for the decision what input is important for the app. The ”Search” is the search for words in libraries. The visualisation of the results of the search belongs to the task ”Display”. There are the questions of what and how to present it. At the end there should be different variations of input: voice record and text input. This is the part of the task ”Input”.

The application should be finished with it’s main functions in the middle of February. This is in the figure 9 marked as ’*App Milestone*’. After this there should only be small implementation tasks and bug fixes.

In the task ’validation’ the software tester and project manager check whether the word assistant app solve the in chapter 2.1 Problem described problem. At the end the software tester has some time for module and system testing of the finished application.

The whole time the project manager takes an eye of all parts of the project. If there would be a customer he could present and investigate the development with them. Finally the project ends with a presentation to defend the thesis.

5.2 Costs

After the arrangement of the timetable for this project, it’s possible to define the budget to develop it. Therefore all direct and indirect costs must considered for the required resources: human, software, hardware resources, an office place, electricity and internet connection.

Direct costs

Human resources The schedule is planned with six different roles and 820 working hours in all. The other time of the project is for the documentation of the this report. The following table 17 shows the estimated costs for the different roles. They are calculated with the the percentage of their working time during the working hours and the normal salary per hour in average.

Salaries normally vary by company and how long someone is working there. The used numbers for the calculation of the salary are median salaries known from [116], [115] and [44] for Germany and Spain.

Role	**Salary per hour**	**Planned working hours**	**Estimated costs**
Project manager	21.42 €	25%/820 = 205.00 h	4391.10 €
Requirements engineer	27.94 €	15%/820 = 123 h	3436.62 €
Systems architect	38.04 €	10%/820 = 82 h	3119.28 €
Designer	21.72 €	10%/820 = 82 h	1781.04 €
Software developer	24.14 €	30%/820 = 246 h	5938.44 €
Software tester	21.71 €	10%/820 = 82 h	1780.22 €
Sum		820 h	20446.70 €

Table 17: Salary per hour for the different roles and how much time was planned for each.

The used **software resources** in this project are all free-ware. Therefore the costs for all used software is 0,00 €. My main IDE was Android Studio (2.3.3). Since the virtual test device there was very slow and sometimes it made problems, I also used Eclipse Neon (4.6) as a second IDE chiefly for testing. For the sketches during the design of the graphical user interface I worked with pen and paper. The drawings in this document are made with WireframeSketcher [157].
The in chapter 3.1 Existing technology described APIs, libraries and licences e.g. for the IDEs programs where free as well. Table 18 shows the costs of software devices for a better overview.

Software	**Costs**
IDE Android Studio 2.3.3	0.00 €
IDE Eclipse Neon (4.6)	0.00 €
WireframeSketcher	0.00 €
Licences, external APIs, Libraries	0.00 €
Sum	0.00 €

Table 18: Costs for software devices

Hardware resources An IDE for the developing is installed on an old ASUS laptop. On this device is a virtual test possible, but the tests are provided on a hardware device during the whole development.
For the project are two hardware devices necessary: a laptop as developing device and an android smart phone to test the application. The used laptop is older than 3 years and hence older than the normal useful life of a laptop. The test device costs less than 410 € after tax, that denote it doesn't count in the writing off. Table 19 shows this costs.

Hardware	**Purchasing price**	**Imputed real costs**
Developing device: ASUS A73SV-TY182V (2011)	625.00 €	0.00 €
Test device: Motorola Moto c plus (2017)	106.10 €	0.00 €
Sum		0.00 €

Table 19: Costs for hardware devices

Indirect costs

Further cost are not directly countable for the project. To this costs belong the internet connection, electricity bill, the price of renting an office place. I don't have a direct office place. I work on the project at my home office in Barcelona. Hence I will handle the calculation like a company that sends it's employee from Germany to Spain.
First of all there are two flights. The flight to Barcelona and the return flight to Germany. That's in total: 135.58 €. In addition, there are higher rental costs in Barcelona than in Hanover and I have to calculate the difference. The rent for the place in Barcelona with 350 € per month is higher than my rent of 250 € in Hanover. This prices include the bills for internet, electricity and gas (heating). That means 100 € more per month and 650 € more for the project duration of six and a half months. I'm an ERASMUS student and get money to compensate this difference. That are 210 € per month which I deduct from the project budget. The remaining costs for living conditions that differ from Germany are not included in the calculation, they must be covered with the salaries calculated with the direct costs which are in table 17.

	Costs per month	**Costs in total (6.5 month)**
Flight	\	135.58 €
Rent		
Barcelona	350.00 €	2275.00 €
-Hanover	- 250.00 €	- 1625.00 €
Difference	100.00 €	= 650.00 €
-ERASMUS	- 210.00 €	- 1365.00 €
Sum		-579.42 €

Table 20: Costs for indirect costs

Risks

It's always possible that not everything in the project runs without problems. For the contingency that something unscheduled happen 15 percent of the direct and indirect costs will be added to the total costs. That are 2718.27 € for this project. In theory it should be possible to finance a second developer or broken hardware with this cache of money.

Cost unit	**Costs**
Costs	
Direct costs	20446.70 €
+Indirect costs	+ -579.42 €
	= 19867.28 €
Risks (15%)	3067.01 €

Table 21: Costs for risk management

Total costs

Cost unit	**Costs**
Direct costs	
Human resources	20446.70 €
+Software resources	+ 0.00 €
+Hardware resources	+ 0.00 €
	= 18701.25 €
+Indirect costs	+ -579.42 €
+Risks	+ 3067.01 €
Total	23513.71 €

Table 22: Costs in total.

The estimated total cost is 23513.71 €.

5.3 Methodology

In general there are different methodologies for a software project possible. From an economic point of view, the best plan would be an accurate plan, which will be implemented in exact that way. From the point of view of programmers, it is clear that this is an impossible wish. Not all problems can be planned right at the beginning. [133]

On one hand this is a research project, but the focus is on the software development. Therefore the methodology is for a software project and I chose the approach of an agile methodology. Agile software development is a methodology that should improve the flexibility of the development and the transparency for all involved persons like the project owner, tester and developer. Thereby it shows faster results than other methologies. The main idea is to keep sub-processes simple and flexible. [149] [129]

Examples for agile development processes are Scrum, Extreme Programming (XP) and Kanban. [149] [129] During my education I worked in projects with Kanban (apprenticeship), was participant in a laboratory to learn the method of extreme programming (studies). During this time I learned more about agile work and the benefits. Reasons to use agile processes and benefits of this kind of process:

- The product delivery is faster because the software is delivered regularly in a working condition for an early usage and review. This working software shows the continuous advance of the project, which motivates the team and supports the morale. [149] [129]

- Thereby the team organizes itself in agile processes and enhance the productivity while it reflects on how it can become more effective and adjusts the process at regular intervals. This additional work of agile development can save time and labour, which in turn enhance the productivity and bring the product to market faster. [133] [149] [129]

- A goal is to get faster results and minimize risks during the process by recognising problems early [133] [149] [129]. This procedure improves the quality and maintainability [149] and the focus is always on a working software. The customer should be made satisfy through this. [129].

- At agile development changes of requirements and priorities are welcome the whole time, even late in development. This changes are used as a competitive advantage because market changes and customer expectations are taken into account. [149] [129]

The first agile method that I consider is Extreme Programming. It is accessible when the customer doesn't have all requirements and adds them during following iterations. For this a work in a team with a customer is necessary but during this master thesis I'm an one person team. That makes the approach of extreme programming impossible. The traditional Scrum with the planned roles makes no sense too.

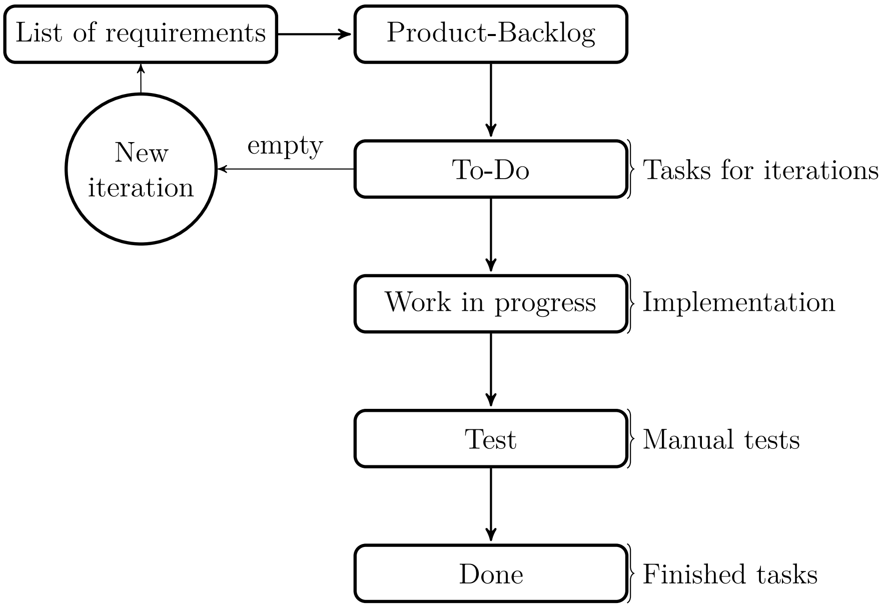


Figure 10: Used Kanban methodology

For this occasion I decided to use Kanban with the work-in-progress method for a single person. This method is flexible for changes when I find new informations and change the requirements after the current iteration. Figure 10 shows the flowsheet of my approach. Every iteration is around 2 weeks long.
At the beginning of this project I started with an analysis of the problem and the purpose to find out e.g. who my stakeholders are, are there different groups with the problem, how is it possible to help this persons. After this I started to write the first requirements and planned the project rough. Than I had an idea of the project and did a literature review of APIs, libraries. This gave me an overview about what already exists. I compared this information with my ideas to interpret what is responsible for this thesis project and split my work into small tasks. [95] With the gained knowledge I checked in each further step the previous findings. For example I searched for similar technologies like voice assistant services and apps which enabled me to refine the previous results.
When my planning looked mature enough, I started with the first Kanban iteration for the implementation. During this process I invented new ideas or found out what isn't working like I thought before. This agile method that I chose made it possible to control and change of the plans at every end of an iteration.
The used Kanban-Board consists of 5 columns: "Product-Backlog", "To-Do", "Work in progress", "Test" and "Done" [36] [120]. The "Product-Backlog" is my collection of requirements for this project. I decided the tasks for the next iteration every second week and put them to the "To-Do" column. At the beginning of an iteration I take one task from "To-Do" and put it to "Work in progress". This is the implementation area followed by manual tests at "Test" to finish the task and put it at the end to "Done". [149] [36] [120]
I didn't do daily stand-ups or similar, because I was working alone. But I reviewed my estimated tasks to optimize the following schedules. When the planning was incorrect, I had some left overs in the "To-Do" area and had to readjust the amount of tasks. The same was necessary when the "To-Do" area was empty and the iteration not over. [149] [36] [120]

6 Technnicality

In chapter 3.1 Existing technology I considered tools and technologies. On one hand I decided what I will use for the development with that knowledge. I describe my decision in chapter 6.1 Technology. On the other hand I specify the requirements for the project. This is written in chapter 6.2 Requirements.

6.1 Technology

In summering up I decide the technology I'm using for this project and explain it here. First I thought about the operating system, programming language and type of the app. The common operating systems for mobile platforms in the marked at this time are iOS (Apple), Android (Google) and Windows Mobile (Microsoft).
I selected Android and therefore Android Studio as development environment. Amongst this I thought about the APIs and Libraries which I explained at the end of chapter 3.1 Existing technology.

6.1.1 Type of app

In my mind it's the best to develop a native app for this project. The reason is to get a high user experience for the word assistant app. Compared with the other types it's the best at a native app. Chapter Existing technology shows that the specific programming languages for the different operating systems are disadvantages of native apps. The specific language for Android is Java, therefore it's no disadvantage for me. Swift for iOS is harder because I have no experiences. Also the other disadvantages of native apps aren't significant e.g. updates in an app store are not planed.
I decided that it should be possible to use the app with a non-continuous good internet connection. It should send so few queries, only as many as necessary for the search. It should still be possible to use the app if the internet connection breaks off and comes back during the speech input. Therefore, the web app type doesn't make sense.
The hybrid type can do it. A cross platform app is not necessary therefore a hybrid with a wrapper is more but unnecessary effort. I decided to build an app only for a single platform and chose android.

6.1.2 Platform

The word assistant app is only for Android, because of the time of this project. In one hand I already have some operating experience with Java and Android. In the other hand it is easier and cheaper to develop and test due the fact that I own an android devices and it's possible to test on virtual devices (Android emulator). More important is the potential to development without specific hardware but free-ware.
In addition to this reasons there are free licences, APIs and libraries for Android. That means it is not out off budget. Some functions are possible without internet connection and supporting more user experience.

API 19: Android 4.4 (KitKat)

Andorid will evolve over time and I had to choose a version for the app. The Android APIs are forward compatible and the lower the api level, the more devices can use the app. The downside is that the latest technologies do not work for the lower api level. When I choose a to old Android API I can't use the current APIs, libraries and new features.
By creating a project in Android Studio the IDE application shows an approximate percentage of how many Google devices the app will be able to run. Google uses the active devices on the Google Play Store for this calculation. For API 19, which is running on Android 4.4 and higher, the assumed number is 90.1 %. This suggests that Android 4.4 aka KitKat is a good choice even when it's announced in 2013. [46]

6.1.3 External APIs and Libraries

Like in chapter 3 State of the Art described there is a Speech To Text API for Android. It lends itself to use these for the word assistant app. The libraries for the search at the word assistant app are TextRazor, the Wordnik API by using the Knicker libraryand the Big Huge Thesaurus. When other languages should be supported the original WordNet database has to be requested and not only this three libraries with English dictionaries.

6.2 Requirements

After planning the project rough I formulate the requirements for this project. This chapter list the important requirements for the word assistant app (in the following text called ”the app”) including task descriptions. It is separated into user experience, functional and non functional (boundary conditions) requirements. It was necessary to decide on the requirements, I explain this decisions after each listing.
For the formulation I used the following pattern which are [37], [74] and [153] describing:

- I use ”Shall” when the requirement has to be in the app.
- With ”Will” I describe facts which are mandatory.
- At least I use ”Should” as well to write about possible extensions which shall be done if there is time left, they are archived goals.

6.2.1 User experience requirements

UE 01	The app shall work with internet connection.
UE 02	The app will understand offline speech input.
UE 03	The app will search online for words.

Table 23: User Experience (UE) requirements for the Word Assistant App.

Table 23 shows the user experience requirements. The reason for UE 01 is that nearly everybody has internet everywhere and the app doesn't need so much storage if it works online instead of an own local database. Therefore the app shall search online for results (UE 3). But it's possible that the internet connection is not optimal, that's the reason why the input will be possible offline (UE 02).

6.2.2 Non-functional requirements

NF 01	The app should work on an android platform (single-platform).
NF 02	The app will work on Android 4.4 and higher (forward compatible)
NF 03	The app will be directly usable. a) The app works without registration. b) The app works without user authentication.
NF 04	The app should save as less data as possible: a) no preferences b) no user content c) no searched words and historical data.

Table 24: Non-Functional (NF) requirements for the Word Assistant App.

The non-functional requirements are in Table 24. This requirements are facts that must be kept. The requirements NF 01 and NF 02 have their seeds in chapter 6.1.2 API 19: Android 4.4 (KitKat). The reason for NF 03 is a good user experience through faster usability. NF 04 concerns the memory area that the required memory should be kept small.

6.2.3 Functional requirements

Table 25 shows the functional requirements for the Word Assistant App. The ”extend” at the left column means this requirements are possible extensions if there is some time for developing left or a follow-up project.

F 01 means that the app should not be published in an app store or commercialized by an other way. It is only developed for this thesis.

Sometimes the area around the user is loud, he doesn’t want or is not allowed to speak because he is e.g. sitting in a library. Therefore the input must be possible without speech. The solution for the word assistant app is requirement F 03. The reason for F 13 is similar. When the user sits in a quite area he disturbs others when his phone e.g. is reading the results loud. Only a silent output makes it possible to use the app everywhere without disturbing people from the area. It would be possible to make a setting to turn the sound on and off, but I have decided against it because the help how to pronounce a word is not a goal of the app .

Requirement F 04 is intended to support the user in usability. If he was misunderstood or fluff during a voice recording, the search input must be correctable and should not be completely reentered.

The end of the input can change the whole search, e.g. when the user finishes the voice input with ”find synonym” (F 10b). Because of this and to keep the data consumption low the app shall start the search only after the input (F 05). The analysis for chapter 3.3 Similar apps showed that the suggestions at other apps are not always helpful. Suggestions are more difficult to implement for voice input in combination with whole sentences. At this app the user can select a type of search by voice input as well. Besides, this he can say different sentences to mixed themes. That makes it hard to propose suggestions especially for the different input kinds and diverse sentences. The app shall not support input with suggestions because this doesn’t improve the user experience a lot and it looks like this effort exceeds the time limit of this work.

I analysed different apps (see chapter 3.3 Similar apps) when I select a result the most of them (B-Rhymes, Offline Thesaurus, Google Translator, Reverso Context, Alternative) the selected search result becomes the new search word. Only some dictionaries give more information or options like e.g. retranslation and audio by clicking on a result. For this reason, I assume that many users are used to this feature and take it as a requirement F 14.

F 15 and F 16 settle on an option to select search filter. A possible selection for search with consideration is to exclude words because of their kind and weight from the search input. Advantages of this is that the user can get faster and more accurate results. Every user has different conditions, therefore they need the feature to work with a changeable

F 01	The app should withheld for this thesis.
F 02	The user shall be able to do the input with speech.
F 03	The app shall be able to do the input with typing.
F 04	The received input should be editable. a) shall by typing. b) shall the entry should be supplemented.
F 05	The app shall start searching results after the input is confirmed a) by the end of voice recording. b) by pressing a search button. c) by changing the search type.
F 06	The input shall be operated in English.
F 07	The input should be possible in English. a) in English.
extend	b) in Spanish. c) in German.
F 08	The app shall search a word that matches to the input.
F 09	The app shall be able a) to find synonyms for words. b) to find antonyms for words. c) to find rhymes for words. d) to find hypernyms and hyponyms for words.
extend	e) to find words for a description by context. f) to translate the input.
F 10	The user should select between the different types of search a) by interface. b) by speech.
F 11	The app shall show the output result of it’s search visual.
F 12	The app should show the search results in a reasonable order.
F 13	The app should be without sounds as feedback or other sound output.
F 14	The search for a selected search result should start after a click on it.
F 15	The app should filter the search to not use a) most frequent words (in English language). b) conjunctions. c) pronouns.
extend	d) other kinds of words.
F 16	The user should be able to select which words are filtered out of the search.
F 17	The app should give the user feedback about working events.
extend F 18	The app should give more informations to results: a) descriptions b) information (e.g. word classes) c) example sentences d) translations

Table 25: Functional (F) requirements for the Word Assistant App.

selection. Possible kind of words are prepositions, conjunctions but verbs, subjects as well. The English language is complex therefore limit the requirement for the app to conjunctions and pronouns. The automatic identification of the child of word goes beyond the the scope of this work, but should be considered as an extension.
In chapter 3.3 Similar apps I examine some good apps for translations (the ranking at the Google Play Store was always higher than 4/5 stars). The biggest difference between them are the supported dictionaries and sizes. Even with an internet community and a project team, it takes a long time to create complete dictionaries and they have to be checked to be correct. I believe this exceeds my language skills to ensure an approaching quality and, in addition to the other requirements of this project, also the scope of the app. Nevertheless, I see it as a good addition to the other features, so a later extension should not be excluded. The same applies to the other ideas of requirement F 18.

7 Design

Based on the specified requirements in the previous chapters I prepare the design of the System and GUI. The description of the most important decisions are described in the chapter 7.1 Specification of the system, it's about the system design and more specific specification. After this I make the design decisions in chapter 7.2 Graphical user interfaces.

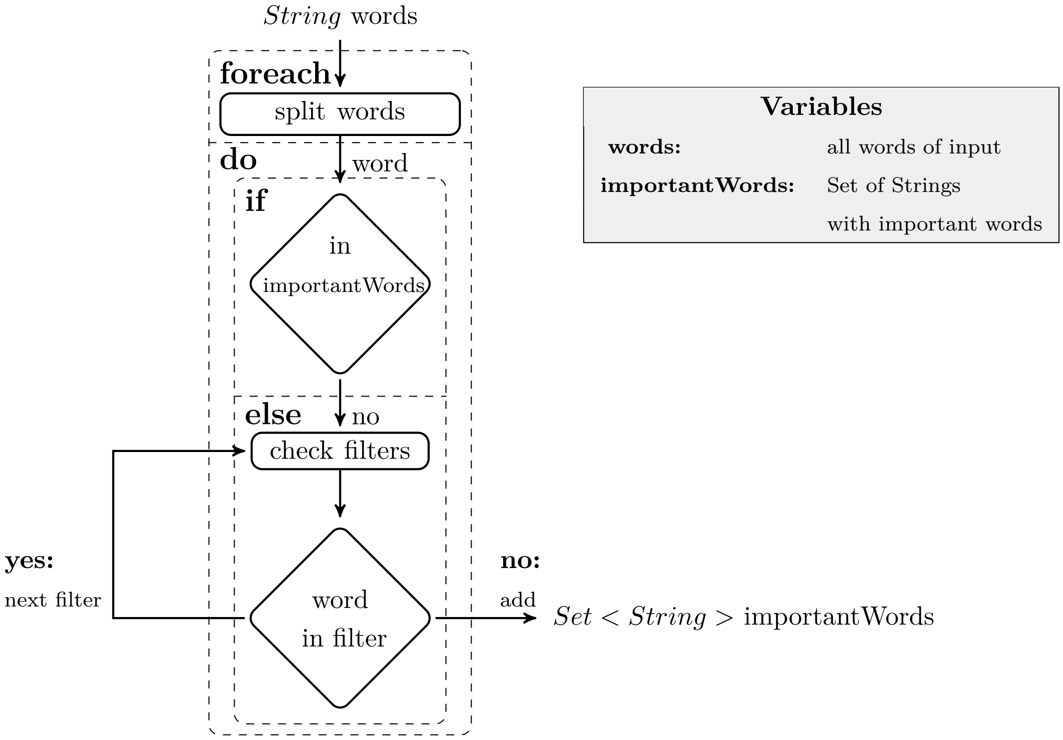
7.1 Specification of the system

I split this chapter in tasks. The first task is which words should be queried. The user can turn on and off filter for this. After this are requests necessary to get results. The last task in this subsection are the weights to sort them.

Filter

The main question for this topic is, which words are less important than others, to find out which are most important for the search. A problem by speaking whole sentences to the app is that there are a lot of common words and filler words at the input. The app has to decide which words are important and which not or less. However, users sometimes just want to use synonymous with, for example, not always using the same filler in the text. Therefore, a filter (F 15) is needed that the users can turn on and off (F 16).
As per [5] all languages, also English, are made up of a small number of very commonly used words, a larger intermediate words and very many rare words. The same source says that 25% of the Oxford English Corpus (huge databank with English of the current and last century [4]) can be formed with the following 10 words in combination with the base form of other words: the, be, to, of, and, a, in, that,have, I.
If we believe the assertion from [34] and [5], the first 25 most used words make up one-third of the written English language. With 100 words half of it is already covered. In [5] they make a difference between the most common words in the written English and ”content words” that regulate the meaning of a sentence and look at them. I need to remove the words without content from the input of the search at the app. Therefore I compared the 100 most frequent used words from [5] and combined them with the most frequent words as per [80], [87], [34] and [5]. My list of most commonly used words in English created this way is in in the appendix The commonest English words.
This list is the first filter. To exclude special word classes I also add filter for conjunctions and pronouns. Every filter is an enum of Strings.

Search important words



Variables	
words:	all words of input
importantWords:	Set of Strings with important words

Figure 11: Diagram to the algorithm to search and extract important words

Figure 11 shows the design for the algorithm of search for important words. The algorithm gets the words of the search input as String and returns a set of important words. It's a set because the order in which the words are entered should not affect the result. Otherwise the result may depend on grammar and sentence structure. Secondly, each keyword is added only once and there are no duplicate words in a set.
First, the input must be split into individual words. All these words will be checked for their importance in a for-loop. When a word is important it has to be added to the set ”importantWords”. The check begins by checking whether the word is already in ”importantWords” or not. If the word already exists in the set, the following filters can be skipped and the algorithm continues with the next word.
When the word is not included it needs to be checked by the filters. This begins with the control which filters are activated. The activation is examined with a boolean. Each filter has an enum with a list of words, if the word matches one of the words of an active filter, the verification is truncated and the next word continues. If the word does not match any of the words in the filter, it will be included in the set ”importantWords”.

Weights of results

Now we have the words which we use for the search and can do server requests to deliver a list of results. This need to shown sorted at the word assistant app. I consider possible opportunities in the following.
Most apps sort the results alphabetically and group them (chapter 3.3 Similar apps). The alphabetically order helps to search and find a word that the user already knows. Furthermore the users understand this kind of sorting easily. But the groups often haven't any header and than it's unclear why they are grouped like they are. Therefore I will not create groups.
Special on the word assistant app is that it collects words of different resources as Strings and can decide which are more significant. The most significant should be at top of the list. Therefore I want to represent how significant a word is by using weights. The higher the weight of a word, the more important it is. When several sources give the same result, it can be assumed that it is a correct result. The probability that the user is looking for this word is greater, hence the weight.
A possible extension could work with a weighting depending on the kind of a word. The problem is: How can I determine which word type should be weighted more than an other. Depending on the situation and the goal, the weighting can be different every time.
In the settings, the user can already exclude groups of words such as pronouns and conjunctions from the words for the search. The user could select a personal weighting at the settings to adapt it to his personal needs as well. E.g. maybe with a multi-level weighting setting. A disadvantage is that he always has to know which setting is necessary to support him the best and an other that he always have to change it for the given situations. This is a lot of effort and bad user experience for him. At the same time he could scroll through all results. That's why the weighting is determined by the frequency.

7.2 Graphical user interfaces

In this chapter I like to present sketches of possible interfaces that I have considered. I made some of theses considerations in comparisons with the apps described in 3.3. In addition to this I describe the ideas to the sketches with their advantages and disadvantages in the following.

Select type of search

"The user should select between the different types of search." (F 10). He only should select only one type at the same time. Other way it's hard to select for every word and a lot effort. Moreover the app should not always ask for the type, the user can select it or use a default selection. The most stakeholder need synonyms chapter 2.1 Problem, therefore this is the default type of search. Figure 12 shows three sketches for this task.

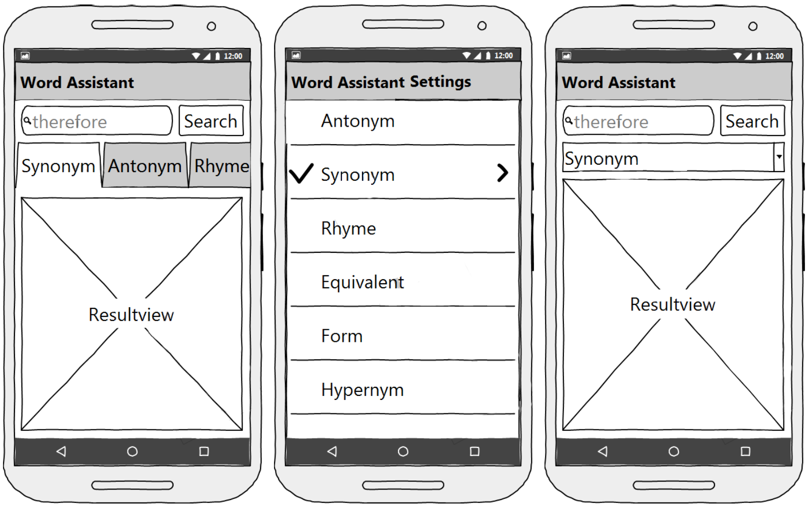


Figure 12: Sketches: Select type of search

The first sketch on the left side shows a tab view. Every type of search could be in a tab of a tab bar. The title of the tab corresponds on the type and needs to be big enough to be readable and clickable with a finger. By selecting a tab, the user would see the corresponding results for the type of search.
When many types are available, the view becomes confusing. One possibility is that the tabs are shown in multiple rows, which fills the screen unnecessarily. A horizontal scrollable list is to be preferred. There it is possible too swipe horizontal through the different taps. A disadvantage is that the user can't see all possibilities.
The settings view is an own view with enough space for a selection. The user opens it at the action bar on top of the main view.
The settings are not visible directly. The user needs to know that it is possible to change the search type and he has to search it because it is hidden. If the user wants to change the search type often this variant is cumbersome to use.
The third variant is a combo box of the types. The figure 12 represent a sketch of this. It always shows the current selection and no others, therefore it is clear how to use. The user knows where to change it and gets a list of all possible selections when he clicks on the combo box.
In consideration of the advantages and disadvantages, the drop down box suites the best. As special feature the chosen type of search is changeable by voice control as well.

Representation of results

Requirement F 12 says ”The app should show the search results in a reasonable order.”, therefore I have to decides how to represent the results that the users understand the order. Figure 13 shows the associated sketches.

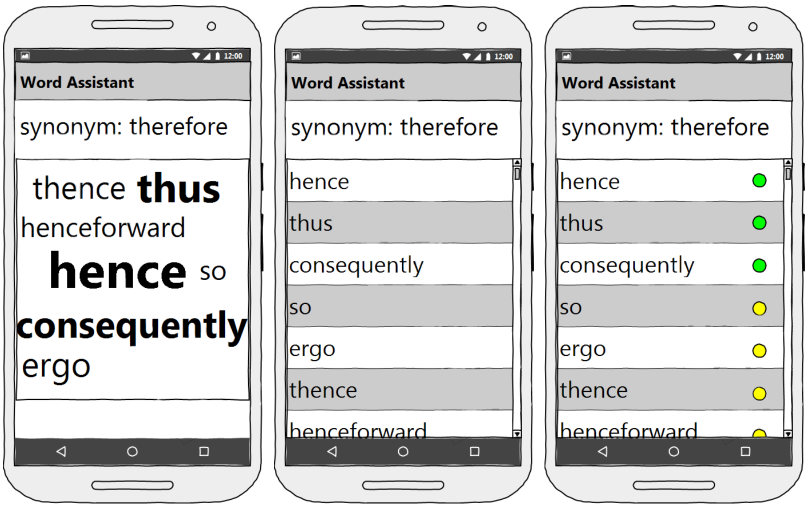


Figure 13: Sketches: how to represent the results

Figure 13 shows three sketches for the representation of results, a word cloud, a list and an extended list with a colour system. A word cloud is known as tag cloud as well. It is a collection of words with different weight. The more important a word is, the bigger it is displayed. The order of the words is possible in different ways, an alphabetically order or a sequence where words with similar weights are next to each other.
By using a word could for the result words the user is presumably locking for the biggest words. It is possible that the app only returns one result. In that case it would have a giant size and unambiguously the user will see it best. But when there are two results are different scenarios possible. Firstly both have the same weight and accordingly the same size. Secondly one word size is a little bigger because it's weight is a little bit bigger. Or in the same example of weights one word is huge and the other small. This depends on the implementation and ensures that the allocation of the weight distribution is always traceable.
The last scenario that I want to explain in this context is a long list of results. Firstly it is possible that the difference between the weights is small and not noticeable. Secondly maybe the view gets overcrowded. In that case a limitation of the displayed results is necessary but than it has to be decided if the user should be able to get the other results or not. This question is not mandatory for e.g. lists.
Compared to the word cloud lists are more commonly used. That means the user understand it directly. An advantage for a long list of results in comparison to word clouds is that there is never a limitation necessary. The user can scroll for more results, or he leaves it. For this the list has to be sorted and shows the with most important first, less important further down the list.
Another option that works well for a list is unlike the Word cloud is that the list can easily show more informations. Some of this information are only bonus but others can support the user to decide which word fits the best for him e.g. a number of importance (weight) or a color system like the app B-Rhymes (see figure 2). All these reasons lead to the decision to implement a list to represent the results of the app.

8 Implementation

Chapter 8.1 Architecture shows the project architecture. Based on the specified requirements I characterize the difficulties and decisions during the implementation in chapter 8.2 Coding. During the development process the application must be tested. I do module tests, integration and system tests which I will not describe in this chapter. This chapter is about the code and implemented algorithms. The algorithms are divided up to search important words, request results, order results and voice control. The final app is presented in chapter 8.3 The user interface where the user interface is introduced.

8.1 Architecture

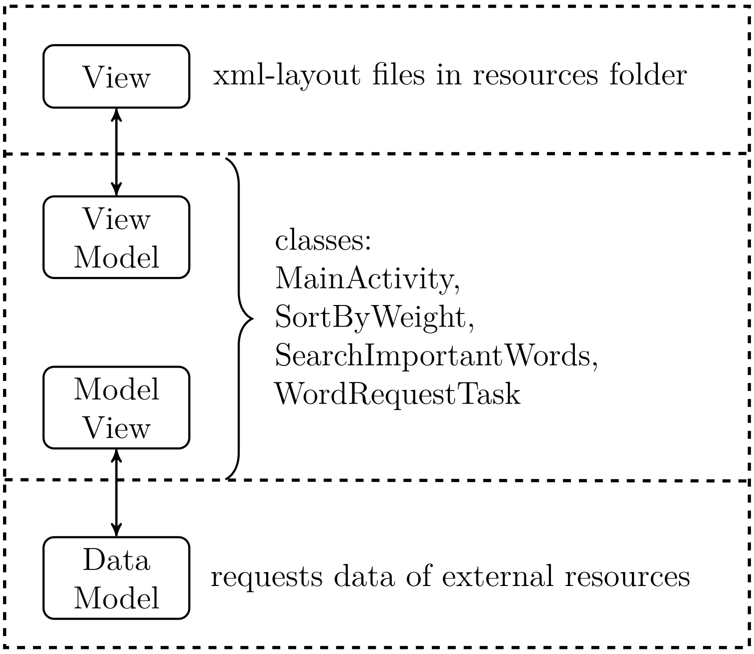

View
xml-layout files in resources folder
View Model
Model View
classes: MainActivity, SortByWeight, SearchImportantWords, WordRequestTask
Data Model
requests data of external resources

Figure 14: Model-View-ViewModel pattern in the word assistant app

I considered different software design pattern for the architecture of the app. The Model View Presenter Pattern with many interfaces is a lot effort for a small app like the word assistant app. This is a reason to use the Model-View-ViewModel Pattern. This is a variant of the famous Model-View-Controller Pattern.

View is the visualisation of the app. The user interface is handled with the xml-layout data in the resource folder. This files can be changed without a change of the other layer, by respecting used identifiers. The View gives the actions of the user to the ViewMode.

ViewModel includes the logic of the user interface. The principal supporter for this is the MainActivity class. It connects the View with the DataModel. It binds properties and instructions for controlling elements of the View. The ViewModel calls methods and services of the DataModel to update the Model or View if necessary.

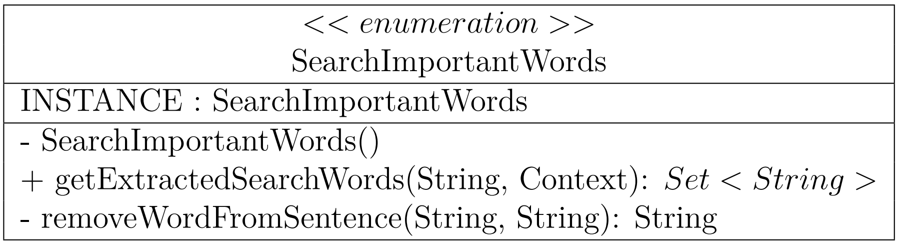
DataModel handles the business logic of the app. This part is important to get and save data for the ViewModel. In this project that are e.g. preference settings and the requested data from external resources.

[109] [110] [111]

8.2 Coding

After deciding the technology, requirements and design this chapter is about the implementation of the algorithms and functions.

Search important words



<< *enumeration* >>
SearchImportantWords
INSTANCE : SearchImportantWords
- SearchImportantWords()
+ getExtractedSearchWords(String, Context): *Set < String >*
- removeWordFromSentence(String, String): String

Figure 15: Enum SearchImportantWords

First of all I implement the voice and text input. The recording of the voice input with the Android's Speech To Text API delivers recognised results in an ArrayList of Strings. The first String in this ArrayList is the result of the recording. To give the user a visual feedback about his voice input this String is shown in a text field (figure 18 at number 2). Besides this the text field is editable, which allows the user to adjust possible mistakes or misunderstood words. The addition of further input is possible by typing in this text field or starting a new recording. The text of the text field affords the String with all words for the search input. For clarity, a line break is inserted after each voice recording.

```java
public Set<String> getExtractedSearchWords(String sentence, Context context){
   Set<String> result = new HashSet<>(Arrays.asList(
      sentence.toLowerCase()
      .replaceAll("(?:[\\.\\;\\,\\:\\!\\?])", "").split("\\s+")));
      .split("\\s+")));
   SharedPreferences sPrefs =
      PreferenceManager.getDefaultSharedPreferences(context);
   String filteredSentence = sentence;
   if (sPrefs.getBoolean(context.getString(
      R.string.preference_mostFrequencyEng_key), false)) {
```

```
11      for (MostFrequencyEng s : MostFrequencyEng.values()) {
12        result.remove(s.getMostFrequencyEng().toLowerCase());
13        filteredSentence = removeWordFromSentence(filteredSentence,
14          s.getMostFrequencyEng());
15      }
16    }
17    ... // repeat for each filter
18    return result;
19  }
```

Listing 1: Code: findImportantWords

The design of the algorithm to extract the important words for the search is explained in chapter 7 Design and figure 11 shows it. Listing 1 shows the implementation of the algorithm.

The parameters of the algorithm are a String and the Context. The Context is the activity in which the function is called. The String includes all words for the search and has to be edited for the extraction of the important words. Before all words are added to a Set of Strings, I change them to be in lower cases and remove the punctuation (point, semicolon, comma, double point, exclamation mark and question mark). Thereafter the String will be separated by white spaces into single words. This all happens in the lines 2 until 5. The method *split()* returns an Array of Strings which I add to a Set of Strings. This removes all duplicates automatically.

To filter the most commonly words, conjunctions and pronouns the algorithm needs the selection of the user. That's what the preference manager of Android takes care of. I initialise the shared preferences in line 6 and 7. In the line after this a String saves the input sentence to filter it together with the Set. The lines 9 to 16 show an example for the filter. All filters have the same structure. Every filter has an unique key and starts with an if-statement. The preference manager returns the selection at the settings by calling this key. If the preference does not exist or can't be found the default is false (line 10).

If the filter is selected: the statement is true and the lines 11-15 will be accessed. For every filter exists an enum of Strings and the algorithm needs to check all enum entries whether the Set contains them or not. If a word of the enum is found it will be removed. In addition the word will be removed from the String filteredSentence with the method *removeWordFromSentence()* (see listing 2).

If the filter is not selected in the preferences the algorithm continuous with the check of the next filter until he run through all filter. At the end the method returns a Set of the Strings which are not removed because of one of the selected filter.

The algorithm is terminating. Because of the different possible filter-selections it is non-deterministic. The same input is always followed by the same result output, which means it is determinateness. It runs in linear time. In the worst case all filters are selected and the algorithm has to check for every of their values if the Set contains this value or not. Therefore time complexity of this algorithm expressed by the big O notation is $O(n)$.

```java
private String removeWordFromSentence(String sentences, String word){
   if (!sentences.toLowerCase().contains(word.toLowerCase())) {
      return sentences;
   }

   String result = sentences.replaceAll(("(?:[\\W])" + word + "(?:[ ])")," ");
   result = result.replaceAll(("(?:[\\W])" + word + "(?:[\\W])"),"");
   result = result.replaceAll(("(?:[\\W])" + word + "(?![\\W\\w])"),"");

   String wordAtBeginning = word.toUpperCase().charAt(0)
      + word.substring(1, word.length());
   result = result.replaceAll(("(?:[\\W])" + wordAtBeginning
      + "(?:[ ])")," ");
   result = result.replaceAll(("(?:[\\W])" + wordAtBeginning
      + "(?:[\\W])"),"");
   result = result.replaceAll(("(?:[\\W])" + wordAtBeginning
      + "(?![\\W\\w])"),"");

   result = result.replaceAll(("(?:[\\W])" + word.toLowerCase()
      + "(?:[ ])")," ");
   result = result.replaceAll(("(?:[\\W])" + word.toLowerCase()
      + "(?:[\\W])"),"");
   result = result.replaceAll(("(?:[\\W])" + word.toLowerCase()
      + "(?![\\W\\w])"),"");

   if (result.startsWith(word)) {
      result = result.replaceFirst((word + "(?:[\\W])"),"");
   } else if (result.startsWith(wordAtBeginning)) {
      result = result.replaceFirst((wordAtBeginning + "(?:[\\W])"),"");
   } else if (result.toLowerCase().startsWith(word.toLowerCase())) {
      result = result.replaceFirst((word.toLowerCase() + "(?:[\\W])"),"");
   }
   return result;
}
```

Listing 2: Code: removeWordFromSentence

As previously mentioned the *removeWordFromSentence()*-method is called for each selected filter at *getExtractedSearchWords()*. Lines 13 and 14 in listing 1 represent this for one of them. The method gets two parameters as input. Both of them are Strings. The first is the sentence from which the other String, a word, should be removed. The sentence is the variable filteredSentence at *getExtractedSearchWords()* and the current entry of the filter is the word to remove from the sentence. *removeWordFromSentence()* stops directly at the beginning and returns the input String if this doesn't contain the searched String. This is not possible before calling the method by checking if the current

entry of the filter removed a String from the Set.
When the word appears at least one time in the sentence the method continue with the lines after line 4 and removes the appearance of the word with regular expressions. The first regular expression ”$(?:[\backslash\backslash W])$” $+ \, word +$ ”$(?:[\ \ ])$” in line 6 sets a white space at the place of the word. It checks with ”$\backslash\backslash W$” that a non-word character like digit or letter is in front of the word and with ”$?:[\ \ ]$” that there is a white space after the word. It is possible that the word is not followed by a white space but a punctuation mark or nothing. Therefore the both next checks examine that the word is followed by a non-word character ”$(?:[\backslash\backslash W])$” or neither a word character nor a non-word character ”$(?![\backslash\backslash W])$”. All this checks grantee that the word is not the beginning or an other part of another word.
The sentence is case sensitive but the word that should be removed not. It is possible that the word starts with an upper case e.g. at the beginning of a sentences. Therefore the word is saved with an upper case as first letter in the variable *wordAtBeginning* in line 10 and thereafter all checks repeated with this String.
By typing fast many people don't care about the right spelling and the words in the filter could start with an upper case or lower case. This is the reason to repeat the replacing in line 19 and the following lines with the word in lower cases.
All checks only remove the word after a white space. When the sentence starts with the word there is either a white space nor a character and the removing needs a special handling. The usage of the *startsWith()*-method in if-statements should support the readability in line 26 and the following lines where the regular expressions remove all three kinds of the word if found. At the end *removeWordFromSentence()* returns the sentence without the word.

Request results

With the Set of important words the search can start. The algorithm for the search is called from the UI thread and runs in an other thread as AsyncTask in the background. AsyncTasks are worker threads to support short operations without blocking the UI thread. When the operation is finished the results are given to the UI where the list view will be refreshed and show them. [48] [51]

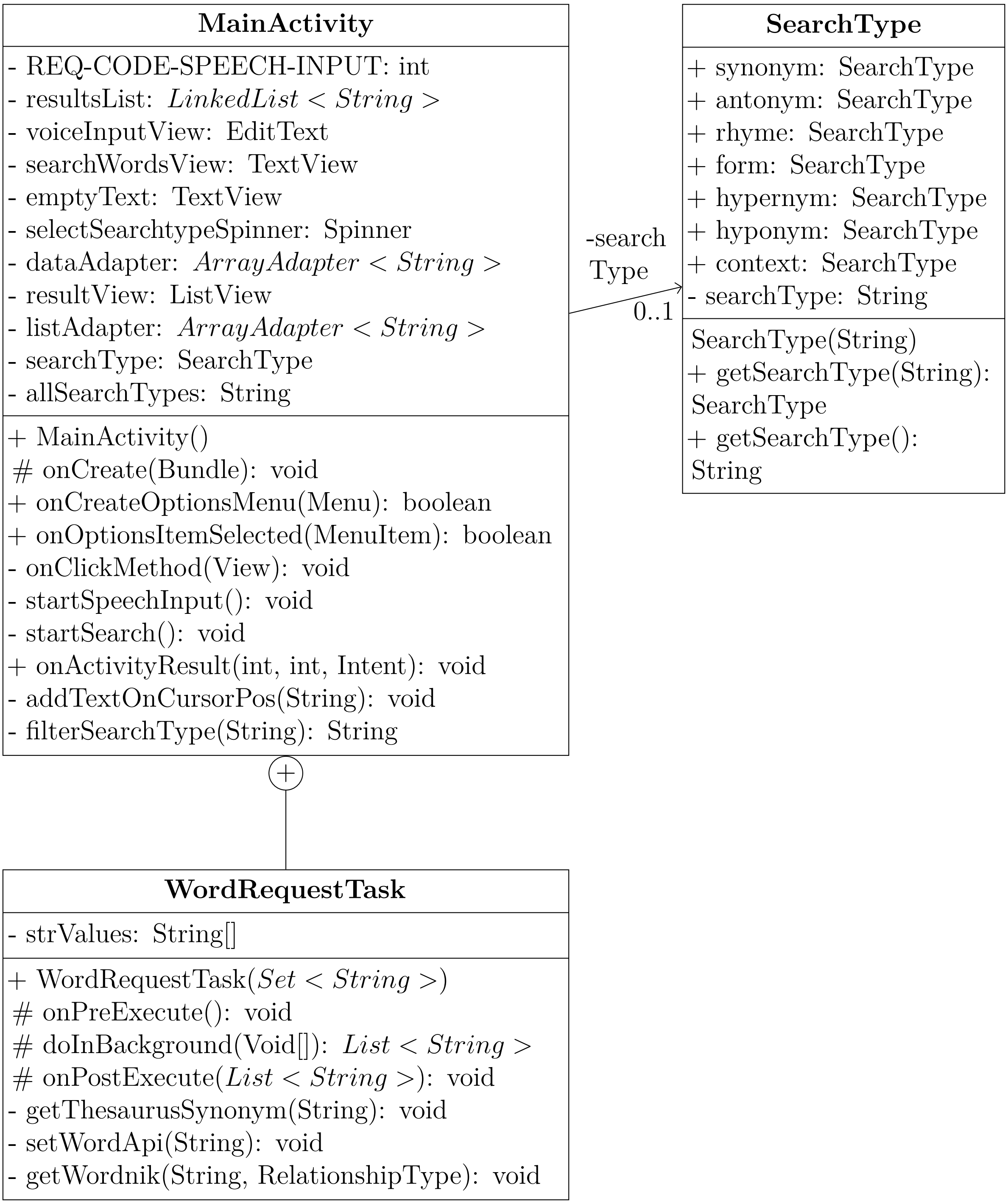

MainActivity
- REQ-CODE-SPEECH-INPUT: int
- resultsList: LinkedList < String >
- voiceInputView: EditText
- searchWordsView: TextView
- emptyText: TextView
- selectSearchtypeSpinner: Spinner
- dataAdapter: ArrayAdapter < String >
- resultView: ListView
- listAdapter: ArrayAdapter < String >
- searchType: SearchType
- allSearchTypes: String
+ MainActivity()
# onCreate(Bundle): void
+ onCreateOptionsMenu(Menu): boolean
+ onOptionsItemSelected(MenuItem): boolean
- onClickMethod(View): void
- startSpeechInput(): void
- startSearch(): void
+ onActivityResult(int, int, Intent): void
- addTextOnCursorPos(String): void
- filterSearchType(String): String
-search Type
0..1
SearchType
+ synonym: SearchType
+ antonym: SearchType
+ rhyme: SearchType
+ form: SearchType
+ hypernym: SearchType
+ hyponym: SearchType
+ context: SearchType
- searchType: String
SearchType(String)
+ getSearchType(String): SearchType
+ getSearchType(): String
WordRequestTask
- strValues: String[]
+ WordRequestTask(Set < String >)
# onPreExecute(): void
# doInBackground(Void[]): List < String >
# onPostExecute(List < String >): void
- getThesaurusSynonym(String): void
- setWordApi(String): void
- getWordnik(String, RelationshipType): void

Figure 16: MainActivity class with innerclass WordRequestTask and associated SearchType

The AsyncTask is a subclass of the main activity. It has an array of Strings for the input words as variable. Figure 16 shows the connection of the Main Activity class with it's subclass WordRequestTask and the association to the SearchType.

```
 1 public class MainActivity extends AppCompatActivity {
 2     private SearchType searchType;
 3     private final LinkedList<String> resultsList = new LinkedList<>();
 4    ...
 5    private class WordRequestTask extends AsyncTask<Void,Void,List<String>> {
 6        private final String[] strValues;
 7 
 8        public WordRequestTask(Set<String> params){
 9            this.strValues = params.toArray(new String[params.size()]);
10            System.setProperty("WORDNIK_API_KEY",
11                "70538348db6b42e43a5181e32070feebc0b303e293ed13a97");
12        }
13 
14        @Override
15        protected void onPreExecute() {
16         ... // prepare UI
17         SortByWeight.INSTANCE.resetMap();
18      }
19 
20        @Override
21        protected List<String> doInBackground(Void... params) {
22            if (strValues != null && strValues.length > 0) {
23                for (String s : strValues) {
24                    setWordApi(s);
25                }
26            }
27            resultsList.addAll(
28                SortByWeight.INSTANCE.getOrderedList());
29            return resultsList;
30        }
31 
32        @Override
33        protected void onPostExecute(List<String> result) { ... }
34 
35      ...// request methods
36     }
37 }
```

Listing 3: Code: AsyncTask as subclass of MainActivity

WordRequestTask(Set<String> params) The constructor instantiates the array of Strings with the words of search which are hand over to the method as input parameter (line 9).

Furthermore it is the place to set up everything for using the libraries e.g. the API key for wordnik by using the system property. The TextRazor will be instantiate only if needed with an API-key (listing 14 on page 108). This depends on the selected type of search. The request for synonyms by Big Huge Thesaurus works without an API key.

onPreExecute() This method is called before the search operation starts. Therefore it prepares the UI with feedback information for the user. The feedback consists of the display of the search words which are the input of the search and sets them as information at the empty text setting of the result view as well. All elements from the list will be removed, therefore the information will be shown until the results can be filled in.

doInBackground(Params...) After the onPreExecute()-method the doInBackground()-method will be called automatically. This accesses the setWordApi(String word)-method for every word that should be searched. Listing 4 shows this method which adds all results to the resultMap which is initialised in line 6.
At the end all collected results have to become sorted by the orderList()-method (listing 6). The method works with the belonging to their weights of each word more about this at page 60.

onPostExecute(Result) This refreshes the results in the UI with it's list adapter.

In the AsyncTask are the different requests e.g. to get synonyms or antonyms. Therefore the selected type of search has to be given. For this the main activity has a variable for the type of search (listing 3 line 2) and every change of the drop down (which is called a spinner in Android) changes the type of this search variable. The object SearchType is an enum with Strings for every possible type (appendix on page 105). As subclass the WordRequestTask as subclass has direct access to this variable.

```
1  private void setWordApi(String word) {
2      switch (searchType) {
3        case hyponym:
4              getWordnik(word, Knicker.RelationshipType.hyponym);
5              new WordTextRazor().getTextRazorHyponym(word);
6              break;
7        // ... other cases
8          case synonym:
9          default:
10             getWordnik(word, Knicker.RelationshipType.synonym);
11             getWordnik(word, Knicker.RelationshipType.equivalent);
12             getThesaurusSynonym(word);
13             new WordTextRazor().getTextRazorSenses(word);
14             break;
15     }
16 }
```

Listing 4: Code: setWordApi() in class WordRequestTask

By using the variable searchType in a switch it is possible to add the different libraries and requests depending to every case. Listing 4 shows the switch with the example of two types of search. One type is the default value which is the case for synonym as well. That case is the only one where the method gets results by the Big Huge Thesaurus web page as well. Most of the results at all cases are collected by the Wordnik API accessed with the Knicker library. So the synonyms as well.
It is possible that the type of search differs in the title by the different libraries e.g. ”equivalent” (Wordnik) and ”senses” (TextRazor). For this reason they are summarized by one of the titles for the search selection. Not only synonym types are bunched together. To support the user with better results and not so much separations in hypernyms with only small differences some groups are created. Therefore the equivalent is requested at synonyms as well.
The second example is the cause ”hypernym”. The requests here are combined and return words for a group of words combined with ”topics”, ”cross reference” and ”Entailment”. This happens in the method *getTextRazorHyponym()* in listing 13.
All types of search in the switch are:

Antonym requests words with the opposite meaning than the input

Context requests related words e.g. words which have the same hyponym as topic are often used in the same context.

Form requests other word formations. This are the primary, inflected and verb form. Also different variants of spellings and words with an identical stem.

Hypernym requests more specific words which represent a topic of the word.

Hyponym requests words that are more specific than the searched word which is their topic. It returns entailments as well, which correspond hyponyms on one-way. In the word assistant app the two-way semantic implication will be returned as hyponym as well even when they are more like synonyms.

Rhyme requests rhymes.

Synonym requests equivalent words and words with the same same understanding.

The method *getWordnik()*, which is on page 106, has the word for the search and the type of the search as parameters. The knicker library returns a list of related objects that include the words for the result. By using two for-loops the method gets this results as String and thereby it calls the *addResult()* for each of them.
The method *getThesaurusSynonym()* (listing 12 at page 106) reads with a BufferedReader the text of $http : //words.bighugelabs.com/api/1/API - Key/" + word + "/$ for the searched word. The synonyms are given line for line and are added to the HashMap with results by using the *addResult()*-method.
The TextRazor supports half of all the search types with own getter methods, the synonyms with senses, the form with stems, the hypernym and the context. The listing

13 represents the *getTextRazorStem()* and *getTextRazorContext()* methods as examples. In order to obtain the corresponding result many for-loops must be run through. The *getTextRazorContext()* adds topics, entailments and the recognized content. The special is that it calls the wordnik method for hypernyms for the result variables of the content request to improve the results.

Order results

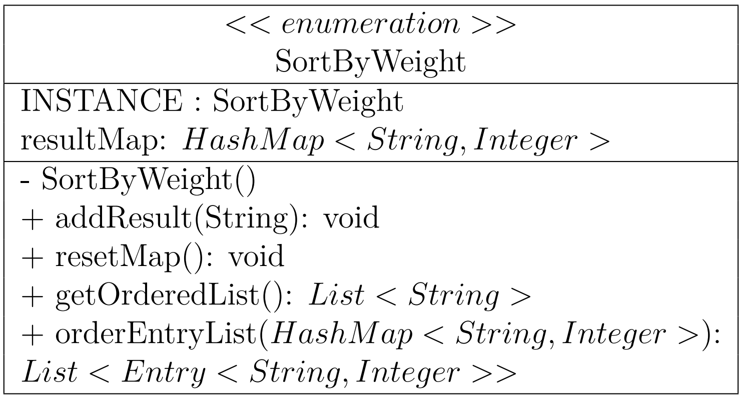

$<<$ *enumeration* $>>$
SortByWeight
INSTANCE : SortByWeight
resultMap: $HashMap < String, Integer >$
- SortByWeight()
+ addResult(String): void
+ resetMap(): void
+ getOrderedList(): $List < String >$
+ orderEntryList($HashMap < String, Integer >$): $List < Entry < String, Integer >>$

Figure 17: Enum SortByWeight

```
private void addResult(String str) {
    if(!str.isEmpty() || str.equals(" ")) {
        if (resultMap.get(str) != null) {
            int i = resultMap.get(str) + 1;
            resultMap.put(str, i);
        } else {
            resultMap.put(str, 1);
        }
    }
}
```

Listing 5: Code: addResults()

As previously mentioned the *addResult()*-method is called for every result independent of the used library and type of search. The used resultMap is a HashMap. Every time when a new search starts the resultMap will be cleared with the *resetMap()*-method (listing 3 line 17). The key of the map is a String and saves a word. Every word in the list of results should be unique therefore it's perfect as key. To every of this words the map saves a value as integer. This integer is the weighting of the words and corresponds on the frequency how often it appeared in the results for all searched words for the current search. Therefore the *addResult()*-method puts all entries in the resultMap. When a key is already used the method increments the value that belongs to the entry, else it puts a new key and value pair.

```java
1  public List<String> getOrderedList() {
2     List list = new LinkedList(resultMap.entrySet());
3     // Defined Custom Comparator here
4     Collections.sort(list, new Comparator() {
5        public int compare(Object a, Object b) {
6           if (((Map.Entry) (a)).getValue()
7                 .equals(((Map.Entry) (b)).getValue())) {
8              return ((Comparable) ((Map.Entry) (a)).getKey())
9                 .compareTo(((Map.Entry) (b)).getKey());
10          } return ((Comparable) ((Map.Entry) (b)).getValue())
11                .compareTo(((Map.Entry) (a)).getValue());
12          }
13       });
14    //copy sorted list
15    ArrayList<String> sortedList = new ArrayList();
16    for (Iterator i = list.iterator(); i.hasNext();) {
17       Map.Entry entry = (Map.Entry) i.next();
18       sortedList.add(entry.getKey() +" ("+ entry.getValue()+")");
19    }
20    return sortedList;
21 }
```

Listing 6: Code: sorting words by their weight

The words with the highest weights are the most important. The background for this statement is on page 48. Listing 6 shows the method that sorts the results for the word assistant app.

I decided to use the sort-method of java.util.Collections, which is using a modified merge sort and has a performance of *O(n log n)*, with an own comparator. The comparator is nothing complicated. First it compares the entries of an HashMap by the value. One difference about this comparator to others is that it sorts backwards to have the higher values at the beginning. Otherwise it had to be turned afterwards because the higher the weight the more important is the word. The second difference is for entries with the same value. If it is the same value the order is alphabetically.

To return a sorted List of Strings, which will be represented in the app, the list will be copied from a LinkedList into a new ArrayList (line 15 and following lines). During the process of copying the key is concatenated with it's weight value in brackets (line 18).

Voice control

It is possible to select a type of search by voice input. This should be easier and faster to use: the user only says a type of search during the voice input and this changes the selected type of search. E.g. he says "search synonym for stop" and the app gives synonyms for "stop" as result and not all four words.

To differ between words that have to be used as input and the words to control the app I designed an algorithm with the following steps:

1. record voice
2. use speech to text for the recording
3. filter text (listing 8)
   - 3.1. if ”find” or ”search”
     - 3.1.1. **yes:** if type of search in input
       - A. **yes:**
         - change type of search
         - remove the filter controlling word (find or search) and the SearchType from the input for the search
         - return input String without removed words
       - B. **no:** return to normal search with previous selections and settings
     - 3.1.2. **no:** return to normal search with previous selections and settings

```
1  public class MainActivity extends AppCompatActivity {
2      private String allSearchTypes = "";
3     ... // other variables and methods
4     @Override
5     protected void onCreate(Bundle savedInstanceState) {
6        ... // initialization stuff
7        for (SearchType st : SearchType.values()) {
8           allSearchTypes += (st.getSearchType() + "|" +
                 st.getSearchType().toLowerCase() + "|");
9        }
10       allSearchTypes = allSearchTypes.substring(0,
            allSearchTypes.lastIndexOf("|"));
11    }
12 }
```

Listing 7: Code: initialization of String allSearchType

Listing 8 is the implemented code for this consideration. It works with regular expressions. The most important String for this is the variable allSearchTypes. This String is initialised in the *onCreate()*-method to always include all SearchTypes. The listing 7 represent how it is initialized. This supports the maintenance when new SearchTypes are added or deleted because nobody needs to change the String.
In line 7 in listing 7 is a for-loop that runs for each entry in the SearchType-enum. Inside this loop are all Strings concatenated twice to the variable allSearchTypes with the regular expression symbol ”|” (for or) as separator. The first concatenation is the SearchType given by the enum and the second changes it into lower cases.

```
 1  private String filterSearchType(String recordResult) {
 2      String regEx = "(find|search|Find|Search)(\\sa\\s|\\s)("
 3         + allSearchTypes + ")(\\sfor\\s|\\s|$)";
 4      Pattern pattern = Pattern.compile(regEx);
 5      Matcher matcher = pattern.matcher(recordResult);
 6      Pattern patternSearchType = Pattern.compile("(" + allSearchTypes + ")");
 7      while (matcher.find()) { // Check all occurrences of regEx
 8          Matcher matcherSearchType =
                patternSearchType.matcher(matcher.group());
 9          if (matcherSearchType.find()) {
10              searchType = SearchType.getSearchType(matcherSearchType.group());
11              selectSearchtypeSpinner.setSelection(dataAdapter.getPosition(
12                searchType.getSearchType()));
13          }
14      }
15      return recordResult.replaceAll(regEx, "");
16  }
```

Listing 8: Code: select type of search controlled by voice

The lines 2 and 3 in listing 8 show one usage of the allSearchTypes-String. The *filterSearchType()*-method gets the recorded text as input and uses the allSearchTypes-String as a part of the regular expression to find out if it is a controller command or normal text input only for the search.

In line 4 the regular expression is compiled a pattern and after this the *pattern.matcher()*-method creates a Matcher with this pattern representation and the given input. The *find()*-method look at all subsequences which matches the pattern by going to the next until the end in a while-loop that runs with this as break condition.

By going through all occurrences of the regular expression a new matcher is created for the second pattern inside the loop. It finds the appearance of the SearchTypes one of the allSearchTypes-variable by using it in a second pattern that is compiled in line 6. The *group()*-method at the matcher returns a subsequence for the last match. If the matcher has one subsequence with a SearchType it will be grouped as well and the searchType-variable of the class changes to it. A subsequence should be found because the first matcher only includes sequences with the control word and SearchType. After changing the the searchType-variable an update of the selection at the drop down (Spinner) is needed.

At the end the first regular expression of this method will be used again. It removes all matches of the controlling words in combination with a type of search from the input String and returns it.

8.3 The user interface

Main view

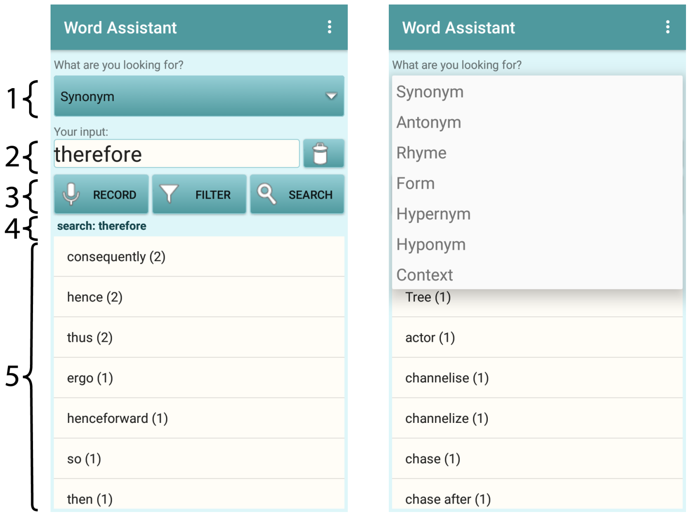


Figure 18: Screenshot: main view

The app has two views: the main view and the settings. The app starts with the main view. The screenshots at figure 18 show this main view.

1) The user can select a search type at the combo box on top of the main view. The screenshot on the right shows the drop down when it is open.

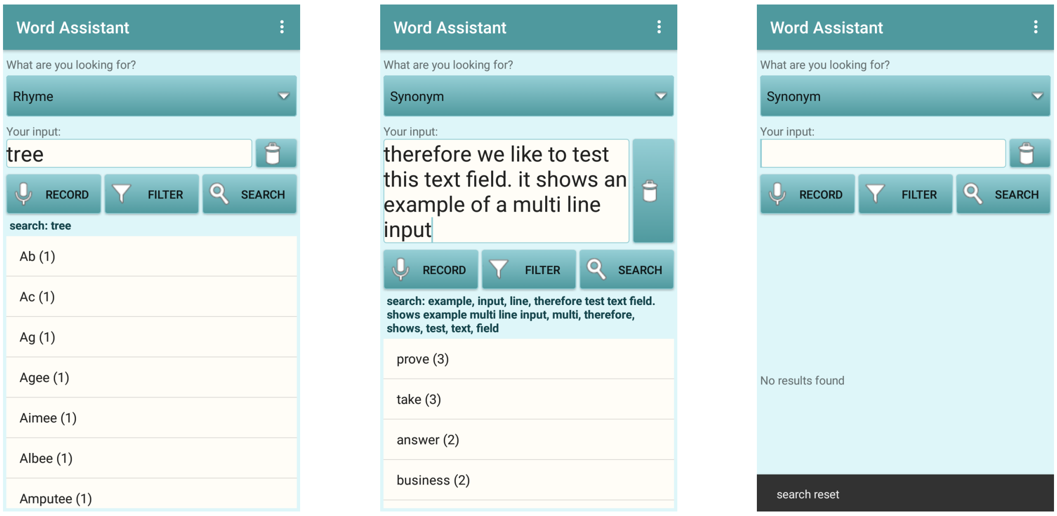
2) All recognized input, voice and text, is shown at this field. It is growing by huge input which is shown at figure 19 in the middle. To the right of the text field is a button with a bin. This can be used to clear the text field and reset the search. The figure 19 shows the main view after pressing this button.

3) This are the buttons to start the voice input, open the filter settings and start the search.

   - After pressing the record button the voice record starts. It ends automatically and thereafter it starts the search.
   - The button with the funnel icon opens the filter settings. Figure 22 shows this view.
   - The search button starts the search with the text at the text field.

4) The used words for the search are represented here.

5) The results of the searches are presented in a scrollable sorted list. Each entry consists of one word and its weighting compared to all results queries. By tabbing on one entry this resets the previous search and the word of the entry becomes a new search input.



What are you looking for? Rhyme

Your input: tree

search: tree

Ab (1)
Ac (1)
Ag (1)
Agee (1)
Aimee (1)
Albee (1)
Amputee (1)

What are you looking for? Synonym

Your input: therefore we like to test this text field. it shows an example of a multi line input

search: example, input, line, therefore test text field. shows example multi line input, multi, therefore, shows, test, text, field

prove (3)
take (3)
answer (2)
business (2)

What are you looking for? Synonym

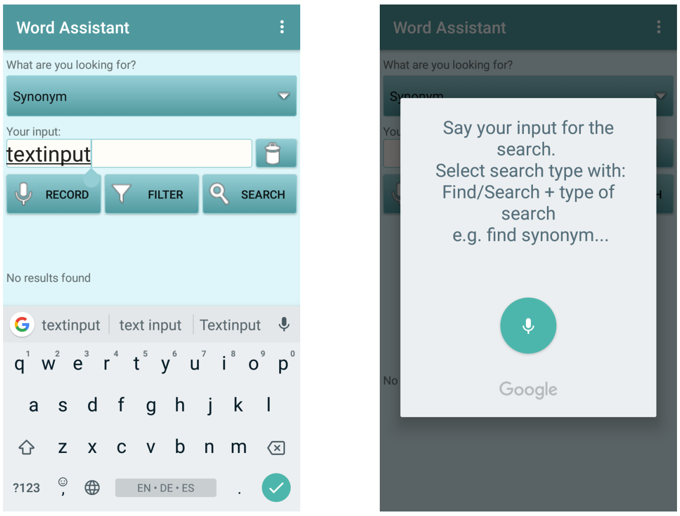
Your input:

No results found

search reset

Figure 19: Screenshot: the main view with selection of rhyme as search type (left), with a huge multi line input (middle), after deleting the input (right)

Search input



What are you looking for? Synonym

Your input: textinput

No results found

Say your input for the search.
Select search type with:
Find/Search + type of search
e.g. find synonym...

Figure 20: Screenshot: text input (right) and voice input record (left)

The input for the search is possible by voice and typing text. Both are shown in figure 20. A third possibility is the selection of a previous result. For the typed input the user can open the soft keyboard by tapping on the text field (figure 18 at number 2). To finish the input the user can close the keyboard with the button with a check mark icon at the corner. The search starts when the input is done or the user presses the search button.

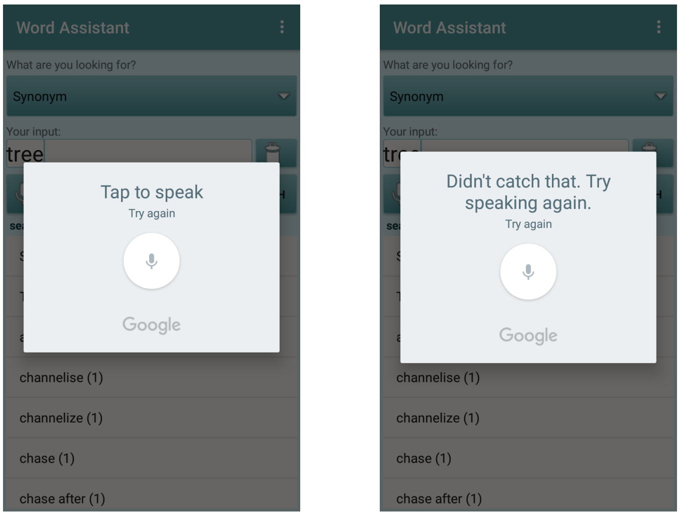
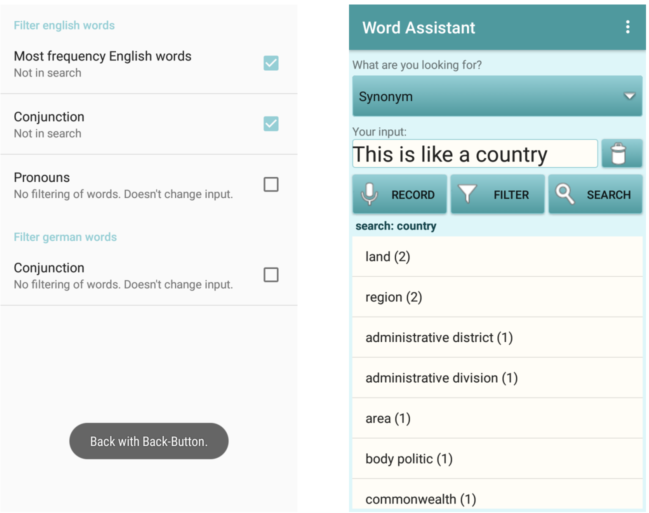


Figure 21: Screenshot: view when the voice input record can't recognize

Like explained in the chapters before the voice recognition can be used to control the selection of the type of search. To inform the user this is explained to the user during voice recording. Figure 21 shows two error messages if the voice input was not understood and at the right the text input. The view changes to the one of this screenshots when the user speaks indistinctly, says nothing or is not understood for any reason. An example for this is a loud environment with many talking people, where it is not possible to use the language input. Therefore, the user is asked to repeat this again.
The user is always able to review the input in a text field for better user experience. He knows what input is understood. If he is misunderstood, he can change the voice input by typing in the text field. Not only a correction is possible, extensions through further entries are possible with both variants.

Selection of filter



Filter english words

Most frequency English words
Not in search ☑

Conjunction
Not in search ☑

Pronouns
No filtering of words. Doesn't change input. □

Filter german words

Conjunction
No filtering of words. Doesn't change input. □

Back with Back-Button.

Word Assistant

What are you looking for?
Synonym

Your input:
This is like a country

search: country

- land (2)
- region (2)
- administrative district (1)
- administrative division (1)
- area (1)
- body politic (1)
- commonwealth (1)

Figure 22: Screenshot: settings view to select the filter

By opening the setting page a toast shows the user how to return to the main page. The left screenshot is taken when the toast was still there. It disappears after a short moment.
The settings page of the app where only pre-set filters can be set is kept simple. Figure 22 shows this simple view at the left picture. The picture on the right is a screenshot of the main view with the selected filter of the other picture. In this example the search runs without ”this”, ”is”, ”like” and ”a”.

9 Verification and validation

Based on the specified requirements in chapter chapter 6.2 Requirements the application can be validated and verified. Module testing is done but not described in this document.

9.1 Verification

Like described in the chapter 5.3 Methodology I did unit and integration tests after every task. A task is only finished when they are successful. Early testing is important and helped to having a better quality and decrease the number of bugs.
At the end of the project I retest the complete app during the system testing. At this testing I compare the app with the defined requirements of chapter 6.2 Requirements. This should ensure that the app satisfy them.

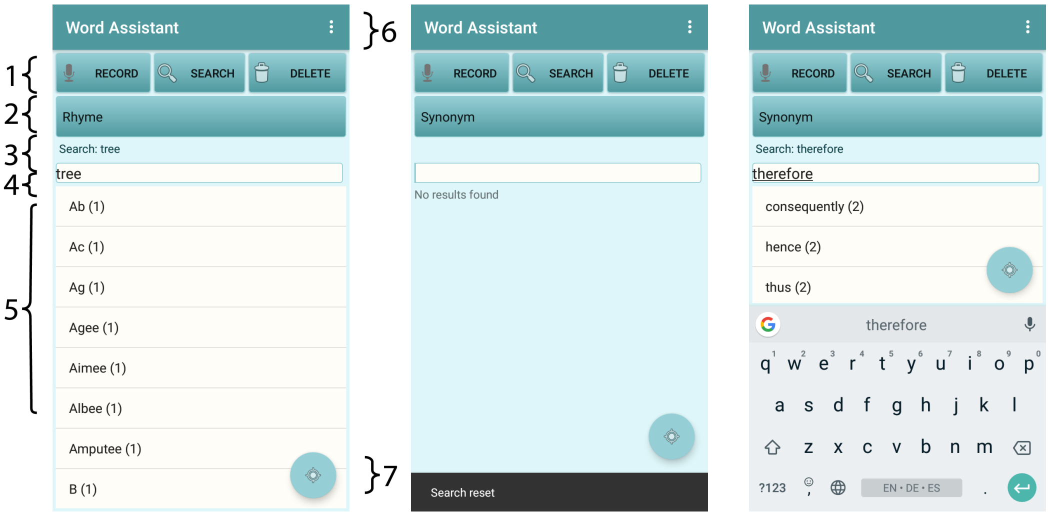
9.2 Validation

The validation deals with the needs of users and the question: Does the word assistant app solve the described problem and helps the users?
The needs of the user belong to the usability and functionality of the word assistant app. To explore this and answer the question as well, I did user tests and analysed their review after a survey. 9 people took part in the survey. For my analysis the users became the app with the task to search a word and get familiar with the app without any descriptions how to use it. After this they answered the following questions of the survey.

1. How easy is the app to use?
   - 1 excellent - 6 worst
2. How easy is the app to understand?
   - 1 excellent - 6 worst
3. Did you find the filter?
   - yes/no
4. Is the filter feature understandable?
   - 1 excellent - 6 worst
5. Would you use the filter?
   - yes/no
6. Do you like the filter?
   - 1 excellent - 6 worst
7. Do you like the voice input?
   - 1 excellent - 6 worst
8. Do you try to change the search type?
   a) by voice?
   b) by tap on it?
9. Which do you prefer?
   - voice, tap, nothing
10. Did the app recognize what you say right?
11. Do you know a situation to use the app? Write it/tell me.
12. Are you satisfied?
    - yes/no
13. Do you have wishes/suggestions/ ideas?
    - free text

9.2.1 UI during survey

After the survey I revised the UI by using the user feedback. Some changes are already mentioned in the list at chapter 9.2.2 Results of survey. The screenshots at figure 23 show the main view during the survey.



Search: tree

tree

Ab (1)
Ac (1)
Ag (1)
Agee (1)
Aimee (1)
Albee (1)
Amputee (1)
B (1)

No results found

Search reset

Search: therefore

therefore

consequently (2)
hence (2)
thus (2)

Figure 23: Screenshot: UI during survey

The arrangement was like the following:

1) The buttons to start the voice input and search were at the top of the view. Together with them the delete button to reset the search input and delete the given input.

2) At the button for the drop down was the down arrow missing. It was looking like all the other buttons. In the final version of the app the selection of the type of search is at the top of the view.

3) Regardless of the survey results, the display of the search words was positioned directly above the search results.

4) The text field for the whole input was at the position where the display of the search words is at the current version. The new positions should improve the understanding. Furthermore the input field and the drop down got labels.

5) The results of the searches were presented as sorted list like now. That one participant answered that he doesn't understand the number in the brackets has to change in a future version.

6) Action bar with the menu button to open the filter settings. One participant of the survey suggested to add a help or about item to the menu. Another comment was that he didn't like to tap two times when there is only one entry to select.

7) This button was an alternative way to open the filter settings. The symbol for the filter remembered to a navigation feature and the position was obfuscating as well. Some users tried to start the search with this.

9.2.2 Results of survey

Usability

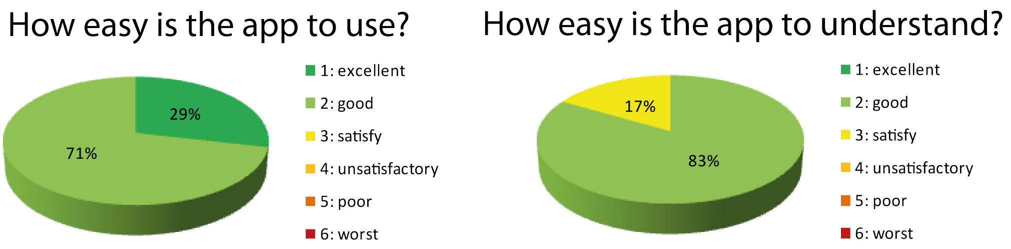


Figure 24: Survey result: usability

The first questions of the survey are about the usability. Most of the participants decided that the app is easy to use and understand. The average of the results at the both first questions is 1.92 and means that the users rate the usability as at least good.

Filter feature

The questions 3-6 refer to the filter feature. Most of the users found the filter direct without any hint. Only two didn't. One of them found the settings after he was told that there is a filter feature somewhere and the other one opened it but didn't notice that this filter-setting was meant with the filter.

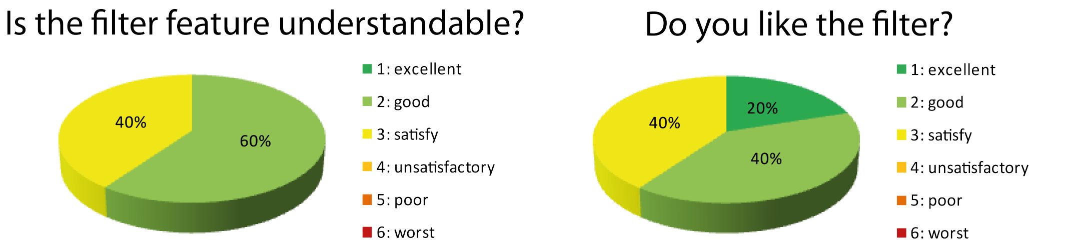


Figure 25: Survey result: filter feature

This influence the next question about the understandability. The result to this question is at figure 25 more than the half of the participants think the filter feature is good to understand. The others think it is satisfying but could be better. Therefore this participants don't want to use the filter feature, but the others (60 %) like to. The rating for the question "Do you like the filter?" is nearly similar to the previous questions. 40 % answered with a 3 and 40 % with a 2 which means "good". The last percent (20 %) answered that they like the filter "excellent".

Voice input

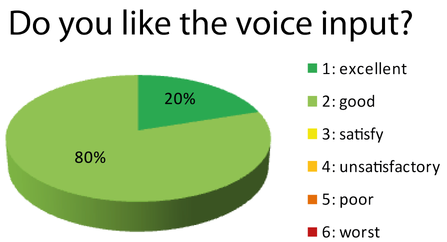


Figure 26: Survey result: voice input

The answers at the questions about the voice input are all very good. All participants evaluate the voice input with at least "good". Most of the time the app recognized the saying correctly.

Selection of the type of search

The questions 8 and 9 are about the selection of the type of search. Figure 27 shows the result of the question which type of selection the participant prefers. The possible answers where voice and tap but some participants had no preference. Therefore I add the option: "both". One of six participants has no preference, all others prefer the selection per voice control.

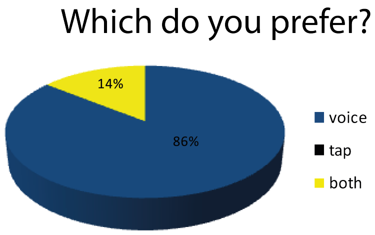


Figure 27: Survey result: selection of the type of search

Others

At question 11. the participants were asked about situations in which the app could be used meaningfully. The following list represents the given answers. Some show that the app solve what it is planned for. Answers like "doing crossword" are new but possible ranges of use.

- study/university, writing stuff for university
- when you want to know a synonym
- writing an email/essay/speech/paper/rap song
- doing crossword
- when you try to explain yourself
- when you try to explain something of your culture in other country
- work e.g. for mails

The participants were satisfied and gave useful feedback. Some suggestions were easy to implement direct and others can't realize during this project and have be be considered in future work. The following list shows the suggestions. The implemented ideas have a check mark (✓). Wishes which are impossible without more research or time have a bullet (•). Ideas that are not desired at this moment got a cross (×).

• shorter recording time

⇒ The recording time depends on the environment. In a loud area (like during this test between a group of talking people) the recording is longer and tries to get more input.

× delete previous input

⇒ Decision in design process: the recording should add and not replace the previous input.

? app doesn't look for phrasal verbs

⇒ depends on the word that the user searches

• I don't get the use of "form"

⇒ my only idea at this moment is to change this is another label for this type of search

• I don't get the numbers in brackets

⇒ The design has to change but it is necessary to consider "how". The problem of using colors is the same like numbers and at the current version most of the time are similar weights at most of the result entries.

✓ the search button should be clearer

⇒ This participant always opened the filter setting instead of starting the search. The suggestion had to be "the filter button should be clearer", that's why this is satisfied. But the position of the button moved more to the border and the symbol of the magnifier has a little bigger shape.

✓ starting the searching process automatically when finished typing

⇒ Changed soft keyboard: no line break but check mark

✓ change filter symbol by a button where is written: "filter" or a funnel-symbol

⇒ Add a button with both to the UI. The funnel-symbol is a good idea.

✓ To show up the filter, cause' of the beginning I think the only use was for synonyms. The meaning?

⇒ Changed the display to change to the filter-setting page. Add the down arrow at the drop down for the selection of the type of search.

× put image

  ⇒ When this belongs to the filter button it is modified ✓. But for the results it is not planned ×.

- if I select e.g. ”fridge”, disable ”form” and ”antonym”

  ⇒ have to be considered in future work

- setting by 3 points (menu) badly that 2 clicks necessary because there is only one option. Better to change direct to ”filter-settings”

  ⇒ that’s right. The idea is to add the item to change the language and/or a help with an instruction of the app.

10 Sustainability analysis

In this chapter I observe sustainability of the word assistant project. At the end of this chapter I will explain my sustainability matrix which is adapted on ideas of Christian Felber [3]. I decided to regard on the whole product life cycle for this and analyse the matrix by it's rows: environmental, economic and social impact. This are the dimensions of sustainability. The columns project development, exploitation and risks belong to this. [38] [3]

Environmental impact

First I take a look on the environmental impact in relation to the project development. To look retrospectively the project already works with only a few resources. It is not possible to use fewer devices but to use devices which consume less energy.

Recharge Smartphone	6 W
Recharge Tablet	8-15 W
Router	7 W
Laptop	50-100 W
Desktop Computer	100-450 W
LCD Monitor	100 W

Table 26: Typical power consumption [27] [154]

The table 26 shows devices that could be useful for the project with their typical power consumption. To reduce the environmental impact of the project I reused existing devices for development and testing. New products have more consequence on the environment because of the removal of the previous devices and the production of the new products. For this reasons the development device is e.g. a laptop instead of a desktop computer.

$$\begin{aligned} best\ case : 50\ watts * 750\ hours &= 37500\ kW = 37.50\ kWh \\ average\ case : 75\ watts * 750\ hours &= 56250\ kW = 56.25\ kWh \\ worst\ case : 100\ watts * 750\ hours &= 75000\ kW = 75.00\ kWh \end{aligned} \tag{1}$$

$$\begin{aligned} best\ case : 100\ watts * 750\ hours + 100\ watts * 750\ hours &= 150000\ kW \\ &= 150.00\ kWh \\ average\ case : 275\ watts * 750\ hours + 100\ watts * 750\ hours &= 243750\ kW \\ &= 243.75\ kWh \\ worst\ case : 450\ watts * 750\ hours + 100\ watts * 750\ hours &= 412500\ kW \\ &= 412.50\ kWh \end{aligned} \tag{2}$$

Another economical benefit is that the typical power consumption of a laptop is less than of a desktop computer and I don't need an external Monitor. The equations 1 and 2 present the calculated kilowatt hours for the project duration for the best, the average and the worst case by using only a laptop (1) and by using the combination of a desktop computer and a monitor (2).

$$60\ wattslaptop * 750\ hours = 45000kW = 45\ kWh \tag{3}$$

The equation 3 shows the calculation of kilowatt hours for the used developing device (ASUS 73SV). The result of 45 kWh is between the best and average case of typical laptop devices (equation 1). The environmental impact can be measured in tons of CO_2. By using the online calculator of [144] the approximate CO_2 emission is around 0.037 tons.

The word assistant app is not using own servers. But more requests have an influence on the economic footprint of this servers as well even if they are operated externally. This makes it more complicated to compare their power consumption with typical servers and I have no control which server the providers use if I want to use their data. The only way to have a positive impact on the economic footstep, was to try to keep the energy consumption of the server and the smartphone as small as possible. Therefore I only send requests to the server if necessary and all actions have to start by an activity of the user like the voice record and recognition. Other way it would waste energy to listen the whole time. The recording ends without a second action, when the word assistant app recognised the input. If the app has problems to get voice input the user gets a message (figure 21) and the recording stops as well.
In comparison to other technical solutions the ecological footprint of the word assistant app is smaller. First it runs on smartphones and the charging of them consumes less power than a tablet or using a laptop or desktop computer (see table 26).
Secondly the users of other solutions have to search on different pages in the internet, various apps or in search engines with a long list of possible results. This takes more time for the search. Therefore it consumes more energy which means it is worst for the environment. The improvement of the word assistant app is that the user should get the results faster and need less electricity. A disadvantage of search engines are not only many results and the decision to choose one. Sometimes people start reading one result and keep reading for longer time.
A possibility is that some people search answers in dictionaries, synonym books and similar books that made of paper. They are still used and don't have a worst ecological footprint even if the user needs several books with a footprint of at least 7.5 kg in average for each book [23]. But if this person owns a smartphone which is able to run the app in addition to the books. The economic footstep of only one of both is less.

Eventualities with environmental impact belong to the environmental risks. The use of devices that consume electricity actually could always counted by it's impact on the environment. Therefore, the use of the app also leads to an increase in power consumption.

In addition the user needs hardware in kind of a mobile phone. But it is implausible that persons start buying mobile devices to use the app. In that case an environmental impact by e-waste is possible to consider but I think it should be very small and is hard to calculate.

Economic impact

The estimated costs of this project are described in chapter 5.2 Costs. I tried to reduce the required budget for this project by keeping the price down and use for example free software if possible. Furthermore I reused existing hardware not only because of environmental impact. Instead of buying new hardware which will be only used for the project I reused existing devices for the development and tests. This saves the money for the initial costs of the project.
Therefore human resources make up the largest part of the cost. At the end of the project it looks like the estimated cost are the same like the final cost. The project runs with a lot of free ware and fix costs for indirect costs like devices and rent.
The only deviation of the cost is possible in the direct cost of human resources. They are hard to determine for this project because the human resource was an only person with different roles and theoretical payment. This payment depend on the different roles because the salary per working hour is between 21.42 € and 38.04 €.

Role	**Salary per Hours**	**Estimated Hours**	**Estimated Costs**	**Working Hours**	**Actual Costs**
Project manager	21.42 €	205.00 h	4391.10 €	218.94 h	4689.69 €
Requirements engineer	27.94 €	123.00 h	3436.62 €	126.28 h	3528.26 €
Systems architect	38.04 €	82 h	3119.28 €	65.60 h	2495.42 €
Designer	21.72 €	82 h	1781.04 €	65.6 h	1424.83 €
Software developer	24.14 €	246 h	5938.44 €	256.66 h	6195.77 €
Software tester	21.71 €	82 h	1780.22 €	86.92 h	1887.03 €
Sum		820 h	20446.70 €	820 h	20221.02 €

Table 27: Calculation of actual costs.

By working the same time the costs change by variations of the plan on the working hours per role. The estimated costs show the assumed plan for the success of the project. They are presented in chapter 5.2 Costs and table 27 sets this costs in contrast with the actual costs. It is hard to quantify the savings because the roles are not clearly separated and not enough documented. By the used methodology is a lot teamwork necessary and every member of the team (the role) can take tasks of other members if necessary and the skills fit. But a possible calculation with changed roles to get the actual costs is in table 27. The difference between the estimated and actual costs are calculated in equation 4. The result is a saving of 225.68 €.

$$20446.70\text{ €} - 20221.02\text{ €} = 225.68\text{ €} \tag{4}$$

It isn't planned to sell the app or commercialize it. The same applies to a possible benefit though self-marketing or as advertisement to sell e.g. services. This makes the project look unprofitable, because the costs of development can not be covered during the exploitation phase. In this case, it is a thesis to obtain a degree, so it is worth the effort and profitable in its own way.
It is possible to continue the project during the exploitation phase, assumed the product life-cycle of the app is longer and doesn't end with this project. In that case the costs of human resources will be less than the first time of this project because the architecture basis, some research and requirements are done. The roles of the systems architect and requirements engineer are the most expensive per hour compared to the others. While continuing the Kanban methodology the software developer can pick the formulated requirements without much effort of the others roles. Furthermore as said in chapter 5.3 Methodology the agile methodology supports changes during the whole life lifetime. That means adjustments and updates are not expensive compared to other features of the app. Another advantage of the Kanban methodology is that errors lead to high maintenance costs later on and other risks are kept as low as possible. As a result, I expect lower costs for the repairs in the exploitation phase. To make the project more feasible from the business point of view and in addition to the considerations mentioned above it should be extended by an other student in another thesis, as student work or a free community project. Without a continuation of the project there is no money planned for reparings etc. but the app would run with the described features of this thesis without causing further costs.

Economic risks are the eventualities to have higher costs than expected because of broken devices or ill developers. The app gets the presented results by external resources. Most of them are requested by a server and it is possible that the provider decide to turn it off some day or change the availability of free API keys and the word assistant app has to pay or use other resources.
A possibility against this risk is to store data on an own server. But than it is necessary to take care of e.g. the installation, the maintenance, the security and the safety.

Social impact

The personal impact on persons that worked on the project is higher than expected at the language skills and ethical. On the ethical level, I didn't expect much change with the project as reason. The research on linguistics and languages is more attentive to listening to people from other ethnic groups. I am paying more attention on the origin of my conversation partner and don't expect the same behave or pronunciation like in my ethnic group. Of course, this is affected and supported by the (Erasmus) environment of the project as well.

The expected influence on the personal level was to consolidate some language skills and feel more comfortable during conversations. Amongst the benefits for later user of the app the improvement was expected by writing the report in a foreign language.
It was essential to read a lot about linguistics and languages during the research. On one hand to consider new features. On the other hand to check the results of the app during the tests. Therefore the upgrading is higher and more language skills than the vocabulary of a foreign language raise up. The professional level includes the personal improvement of conversation skills. The project had significant influence by the usage and consideration of project management techniques e.g. for the planning improved this level. Not to mention I extend my developmental knowledge and consolidate it.

Not only the people who worked on the project and the direct users are effected by the project. Others have advantages by using the word assistant app as well. On one hand the users can help their environment like family and friends by using the app and telling the results. On the other hand people who are reading and listening to texts of people which worked with the usage of the app may profit by this.
The described indirect and passive use has no negative effect. But there is another group: people which are sitting next to an user that is using the voice input in a quite area. In order to avoid or at least reduce this adversely affect, precautionary measures have been taken during the development. In addition to the language input, a text input is possible. The unwanted indirect user can still be disturbed when the user doesn't pay attention or has to use the voice input e.g. if the user does not know the spelling of words or the written input takes too long. Maybe he is in a situation in which he can't type a lot. But negative experience of the group of adversely affected persons is the same like in a situation where the user would ask an other person for help. But the app only shows results and doesn't start a conversation.
Among others the word assistant app should be one possible solution to improve the vocabulary, reduce the problem of blackouts in conversations and help during writing. Some people may improve their language skills easy during a conversation others by reading the words of the app. This depends on the persons and it exists several types of learners. That means the word assistant app is not the only solution for the problems but it is able to solve some. Furthermore the project is not a final application which can't be extended. There are still tasks to improve the app like described in chapter 11.2 Future prospects and the implementation is prepared for extensions.

The word assistant app should support conversations and make it easier to socialize. But there is a risk by using the mobile phone during a conversation for this. The users could be seduced to start doing other things on their phone and loose the interest or track on the conversation.
Another scenario are some lazy users which only use the first result. Sometimes e.g. when a song writer needs rhymes for his work it's necessary to think about the results. It is possible that other results fit better in the context of the song and the wrong selection may changes the story. I would not go that far to say that the app may jeopardize the job of this song writer.

Even when the users read more results they should think about them and remember them if they need this more often. A risk of supporting is to take over the work. That means the user stops working without the support and instead of thinking about other words he will directly use the app.

Sustainability matrix

	Project development (0 to 10)	Exploitation (0 to 20)	Risks (-20 to 0)
Environmental	7	16	-0.5
Economic	4	7	-5
Social	8	12	-8
Sustainability range	43.5		

Table 28: Sustainability matrix

The table 28 illustrates the sustainability matrix for the word assistant app. The project development and Exploitation are rated with positive numbers. The risks can be assigned values between 0 and -20. The 0 means no risks are detected and the -20 means there is a potentially dangerous risks. For all cells count a higher the value is better. A high sum of all cells means a better rating.

The highest possible value at project development is 10. The environmental impact of the word assistant app is estimated with 7. The negative impact is small because of the reused devices but the project can't work without servers, router for internet access, developing and test device. The production and disposal afterwards have impact on the environment. Compared with other projects in this section I evaluate this project as good but with potential to advance.

The reusing of devices keeps the material costs of the project low as well. The encouragement of ERASMUS in combination with a low rent for the indirect costs is a reason that the costs of the project are low in total. Otherwise the project would not be worthwhile without a source of income e.g. selling or as marketing e.g. to get income with other products. Therefore the rating of the project development in view of the economic is only 4 of 10 points.

The project has a big social influence on the persons which are working on it. As only person who is working on the project I have to consider and apply different project management techniques, implement the whole project alone by a schedule that I created before, decided which methodology is the best etc. This and the interdisciplinary research about languages are reasons for the high value of 8.

The reason of writing my master thesis and developing the project for this, would be a reason for a better viability plan at the cell economic/exploitation. But the problem could be solved with other solutions as well. They only need more time and effort. Therefore the value is 7 of 20. But compared to the other solutions the ecological footprint of the word assistant app is better. It saves time and energy during the search because the

users only request what they need and may turn off the phone afterwards. By the use of search engines the may read longer to get the same information or they lose focus and read other things. For this the environmental value at the exploitation phase is very high with 16.
The word assistant app improves the search and so a little bit the quality of life of the user and indirect users. The user is less stressed and saves time because he only needs one app and gets only the requested results. The comfortable way of input effects the quality of life as well. A result of the survey (chapter 9.2.2 Results of survey) was a good usability and that the participants would like to use the app but there are still some improvements possible. On this account I rate this cell as 12.
The risk to increase the economic footprint more than expected is implausible. Therefore the value for environmental risks is only -0.5. The economic risk is not so different. The risk is that the external used server are turned off. The scenarios are more realistic and have a big impact to the results of the project. The size of the effect depends on which server is missing because the other libraries still deliver results. Therefore the value is -5. This is very good considering that the worst is -20. In the hope that people will continue to think about their doing, the risk of losing the job like in the scenario should not be too big. However, we consider the worst cases, so the estimated value is -8.
The maximal sum which are possible to score is 90. The worst case is the minimum with -60. With a final sustainability of 43.5 the project has more than the half of the possible points. This calculation shows that the project is sustainable because of a positive result.

11 Results, future prospects and conclusions

Finally I want to review the whole project. The chapter 5 Project management contains the plan of the project with a schedule for the estimated time and the arising expenses. I compared what worked out and what not turn out well during the project management in chapter 10 Sustainability analysis. The goal of this is to elaborate the applied technique. In this chapter I evaluate the finally result and observe the advantages and disadvantages of the word assistant app which I describe in the chapter 11.1 Results. Out of this I created conclusions for the future. This is described in chapter 11.2 Future prospects. In a final step I evaluate my technique during the project in chapter 11.3 Conclusions.

11.1 Results

First I compare and justify the word assistant app with the functional objectives of chapter 4.2 Objectives. Afterwards I weigh the pros and cons with each other. The conclusion should show the quality of the product and possibilities what could be different. The first functional objective was the decreasing of interruptions during conversations. This is arranged by speech recognition.

The second objective was concerned with a good user experience. In order to check the fulfillment of the second objective, a survey was conducted. The analysis of the results showed that the app can still be partially improved but is already good to use. At the end of the project I already started to improve the app like described in chapter 9 Verification and validation. Further ideas and suggestions are in chapter 11.2 Future prospects.

The last two objectives are related to the creation of the algorithms to decide more important words and order of the results by their importance. The design and the subsequent implementation are applied like described in the chapter 7 Design and chapter 8 Implementation. Their results put the usability of the app together with the UI and is evaluated indirect by the survey.

In addition to the output of search results, the time complexity has been kept as low as possible. As described before e.g. the time complexity of the algorithm to search important words is O(n). Also important for this objectives is that the code is readable for non-project developers and as the case may be subsequent continuation. This supports the maintainable and is less error-prone. In addition, tests had to show that the results of both algorithms are correct in each case and in the interaction together.

11.1.1 Advantages and Disadvantages

In the following I list and describe the advantages and disadvantages of the word assistant app. The plus initiate advantages, the minus stands for disadvantages. The entries with "+/-" are a mix of both and are an advantage and disadvantage at the same time.

+ The word assistant app improves the quality of life of those affected because it save some time and stress when they don't need to switch between different resources by searching certain words.

+ Depending on the user he could feel a bit safer in a conversation and improves his social live because of the help with a few words. But this point of view is very positive and optimistic, it's possible that this is not noticeable.

- A possible negative effect of the last advantage has to be considered but from my point of view it is very improbable at the current product. It is the possibility that an user is to much used too the app and feels uncomfortable when he can't use the app during a conversation.

+ We live in the time of networking in theory therefore its possible to improve the chances of a better career. The app helps by the preparation (learning different words) and by writing an application and motivation letter.

- A problem by using an app with the screen visible during a conversation are notification messages etc. of other apps are allowed by using the word assistant app. A popup by using the app to socialize may have opposite effect.
    - The owner of the phone could see messages to react.
    - The non-owner of the phone is able to read eventual popups.
    - Both persons loose the interest on conversation and start doing things on their phone.

+ When the conversation stopped the app could help. It is possible to use it in a group as well. E.g. by discussing given results or guessing upcoming results as game or thinking about rhymes.

+ The app motivates with a possible game to learn. A famous German educational game is "Stadt-Land-Fluss" where every player needs to find words starting by the same letter for every category as fast as possible. The categories can be changed to the types of search of the app and the user can play against the app or learn to be the best at the next time by playing with his friends. By playing as group the app is a participant and the players try to find other words than the first results to get a higher score or results that appear in the app get double points. The users can decide this by their own.

- By using the word assistant app the user could begin to always search the same words without the need to memorize the results. It is enough to know how to get it and it is easier to ask the app instead of thinking by his own.

+ The app teaches new words and extend the vocabulary of it's user.

    +/- The user doesn't learn a context, examples or other description of the meaning. The app helps most if the user is a native speaker or knows similar words. Sometimes it is necessary to look up some results when using the app with a foreign language and unknown words.

+/- The effect on decreasing misunderstandings as one of the purposes of this work is explained in the text below.

At the end I like to consider the purposes of chapter 2.2 Purpose. During this work I sometimes missed a word or wished to use another option instead of always using the same words. Fortunately I have the app on the testing device and found fast what I was looking for and used this to ameliorate my text.
A misunderstanding is still possible with the app, but in another way than described in the beginning of this work. The misunderstanding in chapter 2.1 Problem is explained by using synonyms or words that only persons with the same background know e.g. local sayings or slang words. The word assistant app is able to look for synonyms and the suggestions can solve the problem and helps understanding. The new problem that I detect is, the need of explanations and the context when a word is used or the information if it is colloquialism or slang word. By learning new words without the context the users may use this words in the wrong way. Consequently a misunderstanding only arises because of this and texts are more colloquial than desired.
Having noticed this and other possibilities for improvement because of the disadvantages, I would like to summarize a possible outlook for future work on this project in the next chapter 11.2 Future prospects. In my opinion, the app helps to find many words that were previously not thought of. For this the purpose ”learning, improve skills, expand vocabulary” is satisfied. But like considered before for non native speaker it may be helpful to get more information about unknown words. The possibility to change the use of the app into an educational game improves language skills and facilitate discussions with other people about the results.

11.2 Future prospects

There is no further planning for this project. It is suitable for further student work in content. In this chapter I'm evaluating possible features for the word assistant app. Some are limited for the current project but possible after more research to other topics or ideas that came to late during this project e.g. only in connection with the survey and user tests. In the following are features for possible extensions considered. Tasks where a closer examination with research is necessary could be possibly potential of further projects.

Other languages In the future the app should support more languages. The current version only provides English. Possible extensions are Spanish and German.

Additional filter The filters should not only provide different languages. In the continuation of the project it's possible to add more filter e.g. to check the kind of a word and only allow results of one word class or forbid one.

Personal filters Another prospect is that the user could add his own filters e.g. when he has own filling words that the app is always using as input. This is not done for the current work because the user can also delete the unwanted words from the input text field. This feature is useful only if the user needs to remove a specific word very frequently.

Context input The first idea of the app was to record a moment of a conversation and get the context. In this current version of the app the results for multiple entries have a higher weighting. The search to get the context works with getting the topic, the input could be improved. As a future prospect this could be improved by the possibility that the user decides whether the search works with single words or the whole input sentences. The used Text Razor API already provides the search of context by sentences and works better with longer inputs.

Translation Looking at the pros and cons, some users have to look up unknown words. The word assistant app should support to make this unnecessary. Therefore the decision against the feature to translate words should be rechecked.

Providing information Like the last prospect one purpose was learning. This could be improved by an extension of the results by giving more information e.g. the word origin, examples to understand in which context the result is normally used or an example sentence. It is necessary to think about which information helps the user, which are to much and confusing and how to present this information in an useful way.

Block popup notifications A possible disadvantage during a conversation are messages of other apps. It is possible to implement e.g. a Notification Listener Service and block notification messages. This must be considered before the implementation.

Disable type of search One participant of the survey had the idea to disable the type of search at the drop down depending on the input. His example was to disable ”form” and ”antonym” when his input is ”fridge”. This idea must be considered in-depth. On the one hand, the question of whether the implementation makes sense. On the other hand, how the implementation is to take place, since the current implementation only searches for results after the selection of the search type and therefore it can’t disable the search type previously.

Representation of weight Another consideration regards the representation of the results. As result of the survey one participant wrote that he didn’t got the meaning of the numbers in brackets. Therefore it should be changed or explained in the next version of the word assistant app.

11.3 Conclusions

In this chapter I look into the technically methods and the personal impact of working on the project. As thesis for the university this project had a time constraint. The project was estimated with 820 hours for the research and implementation by one person. The actual time differs because this documentation was calculated outside the 820 hours. The changes in the schedule of the time and the related costs are discussed in chapter 10 Sustainability analysis.

Because of the interdisciplinary context with languages, which is another environment than computer science, it would have been possible to work in a group with language students and exchange knowledge. Commonly the exchange of knowledge higher by working as a team and the weaknesses of one are balanced by the strengths of others. But then everyone would work on the area he already does best. By my self-directed work the learn effect in this project by analysing, schedule and implement without a team was high and helped to understand the complexity of the typical projects. Especially with regard to the sustainability and social commitment.

The technique of the sustainability analysis was new for me but helped to structure some thoughts and measure them with values. Furthermore it contains considerations on economic, social and ethical areas which I wouldn't pay this much attention without this. The analysis helped to structure the work, which is important for technical work. Thereby it is necessary to evaluate the analysed informations and how it is possible to visualise it for others in a comprehensible way.

Besides the planning and documentation this project has a developing part to create an innovation. For this I chose a kind of Kanban-methodology for the development. This facilitated a continuous improvement by reviewing my work after every period. It motivates to see the progress like in a real working project and I was flexible for technical changes when I e.g. found a new library.

To sum up, I'd like to say that this project had an improvement on many different skills. Professional and personal skills as well. A view on the professional skills shows first an improvement of language skills with background knowledge and to apply in conversations or writing. Secondly I strengthened my developmental knowledge. The biggest growth concerns my project management skills during the autonomous work.

Bibliography

[1] Appendix: False friends between English and Spanish. https://en.wiktionary.org/wiki/Appendix:False_friends_between_English_and_Spanish. Accessed: 2018-01-10.

[2] Vocabulary. https://en.wikipedia.org/wiki/Vocabulary. Accessed: 2018-03-14.

[3] Guidelines for the Sustainability Report of the Final Degree Project. https://www.upc.edu/, 2016.

[4] 2016 Oxford University Press. The Oxford English Dictionary. https://www.oxforddictionaries.com/oed. Accessed: 2018-02-19.

[5] 2016 Oxford University Press. What can the Oxford English Corpus tell us about the English language? https://en.oxforddictionaries.com/explore/what-can-corpus-tell-us-about-language. Accessed: 2018-02-19.

[6] Amazon Inc. Amazon Echo - Black (1st Generation). https://www.amazon.com/Amazon-Echo-Bluetooth-Speaker-with-WiFi-Alexa/dp/B00X4WHP5E. Accessed: 2018-02-10.

[7] Amazon Inc. Amazon Echo Dot (2. Generation), Schwarz. https://www.amazon.de/Amazon-Echo-Dot-Generation-Schwarz/dp/B01DFKBG54. Accessed: 2018-02-10.

[8] Amazon Inc. Amazon Echo, Schwarz (Vorherige Generation) . https://www.amazon.de/Amazon-SK705DI-Echo-Schwarz/dp/B01GAGVCUY. Accessed: 2018-02-10.

[9] Ambar Ideas. No Lo Digas. https://play.google.com/store/apps/details?id=com.ambar.ideas.nolodigas. Accessed: 2017-09-30.

[10] Amy Ann Forni, L. G. Gartner Says Worldwide Sales of Smartphones Grew 7 Percent in the Fourth Quarter of 2016. https://www.gartner.com/newsroom/id/3609817. Accessed: 2017-11-5.

[11] Apple Inc. Jump Right In. https://developer.apple.com/library/content/referencelibrary/GettingStarted/DevelopiOSAppsSwift/. Accessed: 2018-02-10.

[12] Apple Inc. Verfügbarkeit von ios features. https://www.apple.com/de/ios/feature-availability/. Accessed: 2017-09-21.

[13] Apple Inc. What's New in Xcode 9. https://developer.apple.com/xcode/. Accessed: 2018-02-10.

[14] Apple Inc., Lizenz: Creative Commons Attribution - Share Alike. ”Hey Siri”. https://www.apple.com/ios/siri/. Accessed: 2017-09-21.

[15] Apple Inc., Lizenz: Creative Commons Attribution - Share Alike. ”Hey Siri” German. https://www.apple.com/de/ios/siri/. Accessed: 2017-09-21.

[16] Apple Inc., Lizenz: Creative Commons Attribution - Share Alike. ”Hey Siri” Spanish. https://www.apple.com/es/ios/siri/. Accessed: 2017-09-21.

[17] Armour, B. iOS vs. Android: Which Platform Should You Build For? https://clearbridgemobile.com/ios-vs-android-app-development-which-platform-should-you-develop-for/. Accessed: 2017-11-5.

[18] Arputharaj, V. Android Speech to Text Tutorial. http://stackandroid.com/tutorial/android-speech-to-text-tutorial/. Accessed: 2017-12-05.

[19] Arputharaj, V. How to enable offline Speech To Text in Android. https://stackandroid.com/tutorial/how-to-enable-offline-speech-to-text-in-android/. Accessed: 2017-12-05.

[20] Ayush Gupta, Ivan Porto Carrero, E. M. About Wordnik. http://www.wordnik.com/about. Accessed: 2018-02-10.

[21] Ayush Gupta, Ivan Porto Carrero, E. M. developer.wordnik.com. http://developer.wordnik.com/. Accessed: 2018-02-10.

[22] Bager, J. Google Home in Deutschland: Eine ernstzunehmende Konkurrenz für Amazons Echo. https://www.heise.de/newsticker/meldung/Google-Home-in-Deutschland-Eine-ernstzunehmende-Konkurrenz-fuer-Amazons-Echo-37 html. Accessed: 2018-01-05.

[23] Carpenter, M. Books vs ebooks: Protect the environment with this simple decision. https://theecoguide.org/books-vs-ebooks-protect-environment-simple-decision. Accessed: 2018-04-14.

[24] Christian M. Meyer, Jan Berkel, Y. Y. T. M. I. V. S. Java Wiktionary Library. https://github.com/dkpro/dkpro-jwktl. Accessed: 2018-02-10.

[25] CMU Sphinx. PocketSphinx on Android. https://cmusphinx.github.io/wiki/tutorialandroid/. Accessed: 2017-12-05.

[26] Cosmicode. GuessUp Party Charades. https://play.google.com/store/apps/details?id=pt.cosmicode.guessup. Accessed: 2017-09-30.

[27] DaftLogic. List of the Power Consumption of Typical Household Appliances. https://www.daftlogic.com/information-appliance-power-consumption.htm. Accessed: 2018-04-14.

[28] de Vries, R. How To Develop iOS Apps On A Windows PC. https://learnappmaking.com/develop-ios-apps-on-windows-pc/. Accessed: 2018-02-10.

[29] Dictionary.com, Llc. Thesaurus.com. http://www.thesaurus.com. Accessed: 2018-02-09.

[30] divya. Android Speech To Text Tutorial - Part1. http://www.techjini.com/blog/android-speech-to-text-tutorial-part1/. Accessed: 2017-12-05.

[31] Dua, K. A Guide to Mobile App Development: Web vs. Native vs. Hybrid. https://clearbridgemobile.com/mobile-app-development-native-vs-web-vs-hybrid/. Accessed: 2018-03-10.

[32] e-Zest Solutions. Mobile platforms, frameworks & environments. http://www.e-zest.com/mobile-operating-system/. Accessed: 2018-02-10.

[33] Editorial office The Local. Top ten funniest Spanish-English false friends. https://www.thelocal.es/galleries/culture/funny-false-friends. Accessed: 2018-01-10.

[34] Edward Bernard Fry, Ph.D, J. E. K. E. . D. L. F. E. *The Reading Teacher's Book of Lists*. J-B Ed Series. Prentice Hall, 2000.

[35] Eichfelder, M. Google Home vs. Amazon Echo: Wer ist der bessere Assistent? http://www.chip.de/news/Google-Home-vs.-Amazon-Echo-Die-Assistenten-im-Vergleich_120738644.html11. Accessed: 2018-01-05.

[36] Faber, J. Kanban - Produktivität für Teams und Einzelkämpfer. https://simplizist.de/kanban-einzelperson/. Accessed: 2018-01-30.

[37] Famuyide, S. 15 Tips For Writing Better Requirements. https://businessanalystlearnings.com/blog/2013/7/26/15-tips-for-writing-better-requirements. Accessed: 2017-10-10.

[38] Felber., C. *References u.a. C. Felber. Die Gemeinwohl-Oekonomie - Das Wirtschaftsmodell der Zukunft. 2010.* Piper, 2010.

[39] Floemer, A. Die Zukunft der Suche: Das kann der Google Assistant: Google Assistant: Auch für Android TV, Android Wear 2.0, Android Auto und andere Geräte. https://t3n.de/news/google-assistant-754347/2/. Accessed: 2018-01-05.

[40] Floemer, A. Die Zukunft der Suche: Das kann der Google Assistant: Google Assistant: Die Zukunft der Suche ist individuell. `https://t3n.de/news/google-assistant-754347/`. Accessed: 2018-01-05.

[41] (for FluentFlix), L. 20 Surprising Spanish-English False Friends Everyone Falls For. `https://www.fluentu.com/blog/spanish/spanish-english-false-friends-cognates/`. Accessed: 2018-01-10.

[42] Forsberg, M. Why is Speech Recognition Difficult?

[43] gamesnapps4u. Offline Thesaurus Dictionary. `https://play.google.com/store/apps/details?id=de.offlinethesaurus`. Accessed: 2018-01-30.

[44] Glassdoor Inc. ... Salaries. `https://www.glassdoor.com/Salaries/...` Accessed: 2017-09-30.

[45] GNDZ. Synonyme (Alternative). `https://play.google.com/store/apps/details?id=tgz.synonyme`. Accessed: 2017-09-30.

[46] Google Ilc. Android 4.4 APIs. `https://developer.android.com/about/versions/android-4.4.html`. Accessed: 2018-02-10.

[47] Google Ilc. android.speech. `https://developer.android.com/reference/android/speech/package-summary.html`. Accessed: 2017-12-05.

[48] Google Ilc. AsyncTask. `https://developer.android.com/reference/android/os/AsyncTask.html`. Accessed: 2018-01-08.

[49] Google Ilc. Google Play. `https://play.google.com/store`. Accessed: 2017-09-30.

[50] Google Ilc. Google Übersetzer. `https://play.google.com/store/apps/details?id=com.google.android.apps.translate`. Accessed: 2017-09-30.

[51] Google Ilc. Processes and Threads Overview. `https://developer.android.com/guide/components/processes-and-threads.html`. Accessed: 2018-01-09.

[52] Google Ilc. Settings. `https://developer.android.com/guide/topics/ui/settings.html`. Accessed: 2017-12-11.

[53] Google Ilc. Settings. `https://material.io/guidelines/patterns/settings.html`. Accessed: 2017-12-11.

[54] Google Ilc. SpeechRecognizer. `https://developer.android.com/reference/android/speech/SpeechRecognizer.html`. Accessed: 2017-12-05.

[55] Google Inc. Google Home. `https://play.google.com/store/apps/details?id=com.google.android.apps.chromecast.app`. Accessed: 2018-02-10.

[56] Google Inc. Google Now - Die richtigen Informationen zur richtigen Zeit. https://www.google.com/intl/de/landing/now/. Accessed: 2018-02-10.

[57] Google Inc. Google Now Launcher. https://play.google.com/store/apps/details?id=com.google.android.launcher&hl=de. Accessed: 2018-02-10.

[58] Google Inc. Google Now Launcher. https://play.google.com/store/apps/details?id=com.google.android.launcher&hl=en. Accessed: 2018-02-10.

[59] Google Inc. Google Now. The right information at just the right time. https://www.google.com/intl/en-GB/landing/now/. Accessed: 2018-02-10.

[60] Google Inc. Implementing a Custom Request. https://developer.android.com/training/volley/request-custom.html. Accessed: 2018-02-10.

[61] Google Inc. Making a Standard Request. https://developer.android.com/training/volley/request.html. Accessed: 2018-02-10.

[62] Google Inc. RecognizerIntent. https://developer.android.com/reference/android/speech/RecognizerIntent.html. Accessed: 2017-12-05.

[63] Google Inc. Sending a Simple Request. https://developer.android.com/training/volley/simple.html. Accessed: 2018-02-10.

[64] Google Inc. Setting Up a RequestQueue. https://developer.android.com/training/volley/requestqueue.html. Accessed: 2018-02-10.

[65] Google Inc. Transmitting Network Data Using Volley. https://developer.android.com/training/volley/index.html. Accessed: 2018-02-10.

[66] Grundner, A. Android App Entwicklung mit Eclipse, SDK, ADT. http://greenitsolutions.at/android-eclipse-sdk-adt-turorial-deutsch/. Accessed: 2018-02-10.

[67] Gupt, M. Android Volley Example. http://www.truiton.com/2015/02/android-volley-example/. Accessed: 2018-02-10.

[68] Gupt, M. Truiton/VolleyExample. https://github.com/Truiton/VolleyExample. Accessed: 2018-02-10.

[69] Heiges, N. False friends: embarrassed or embarazada? http://www.cristinacabal.com/?p=291. Accessed: 2018-01-10.

[70] Hemetsberger, P. dict.cc Wörterbuch. https://play.google.com/store/apps/details?id=cc.dict.dictcc. Accessed: 2017-09-30.

[71] Herman, J. 50 Spanish-English False Friend Words. http://mentalfloss.com/article/57195/50-spanish-english-false-friend-words. Accessed: 2018-01-10.

[72] III, G. N. R. 10 Most Important Business Objectives. http://smallbusiness.chron.com/10-important-business-objectives-23686.html. Accessed: 2017-09-30.

[73] Iris Fostiez, Johan De Bruyn, S. B. How can I calculate the consumption of an electrical appliance? https://www.energuide.be/en/questions-answers/how-can-i-calculate-the-consumption-of-an-electrical-appliance/94/. Accessed: 2018-04-14.

[74] James, A. Business Requirements: Should you use Shall v Will? http://klariti.com/business-writing/business-requirements-should-you-use-shall-v-will/. Accessed: 2017-10-10.

[75] Johansson, S., Hasselggård, H., and Oksefjell, S. *Out of Corpora: Studies in Honour of Stig Johansson*. Language and computers : Studies in practical linguistics. Rodopi, 1999.

[76] Jurran, N. Google Assistant: Verschärfter Angriff auf Amazons Alexa. https://www.heise.de/newsticker/meldung/Google-Assistant-Verschaerfter-Angriff-auf-Amazons-Alexa-3850021.html. Accessed: 2018-01-05.

[77] Jurran, N. Sprachassistenten: Gleich drei neue Lautsprecher mit Google Assistant. https://www.heise.de/newsticker/meldung/Sprachassistenten-Gleich-drei-neue-Lautsprecher-mit-Google-Assistant-3817809.html. Accessed: 2018-01-05.

[78] K, N. K. Choosing Single or Cross Platform and the Ideal Mobile Development Techniques. https://www.tavant.com/blog/choosing-single-or-cross-platform-and-ideal-mobile-development-techniques. Accessed: 2018-02-10.

[79] Kalenda, F. Google Assistant: Entwickler können sich ab Anfang Dezember integrieren. https://www.zdnet.de/88280159/google-assistant-entwickler-koennen-sich-ab-anfang-dezember-integrieren/. Accessed: 2018-01-05.

[80] Kelk, B. Top 1000 Words. http://www.bckelk.ukfsn.org/words/uk1000n.html. Accessed: 2018-02-19.

[81] Klosowski, T. I Want to Write iOS Apps. Where Do I Start? https://lifehacker.com/i-want-to-write-ios-apps-where-do-i-start-1644802175. Accessed: 2018-02-10.

[82] Klubnikin, A. Cross-platform vs. Native Mobile App Development: Choosing the Right Dev Tools for Your App Project. https://medium.com/all-technology-feeds/cross-platform-vs-native-mobile-app-development-choosing-the-right-dev-tools-fo Accessed: 2018-02-10.

[83] Laven, K. 5 Benefits of Managed Voice Services. https://itel.com/5-benefits-managed-voice-services/. Accessed: 2018-02-10.

[84] Lenzo, K. The CMU Pronouncing Dictionary. http://www.speech.cs.cmu.edu/cgi-bin/cmudict. Accessed: 2018-02-10.

[85] LEO GmbH. LEO. https://dict.leo.org/. Accessed: 2017-12-12.

[86] LEO GmbH. LEO Wörterbuch. https://play.google.com/store/apps/details?id=org.leo.android.dict. Accessed: 2017-09-30.

[87] Leon Dixon, B. G. The First 100 Most Commonly Used English Words. http://www.duboislc.org/ED-Watch/Words/1-100.html. Accessed: 2018-02-19.

[88] Li, A. Google has created offline voice recognition that is 7x faster than an online system. https://9to5google.com/2016/03/11/google-accurate-offline-voice-recognition/. Accessed: 2017-12-05.

[89] Lin, M. B-Rhymes Android App. http://www.b-rhymes.com/android/. Accessed: 2017-09-30.

[90] Lin, M. Find Words That Almost Rhyme. http://www.b-rhymes.com/. Accessed: 2017-09-30.

[91] Link, M. Apple kauft Firma für künstliche Intelligenz. https://www.heise.de/newsticker/meldung/Apple-kauft-Firma-fuer-kuenstliche-Intelligenz-3713648.html. Accessed: 2018-01-05.

[92] LoveToKnow, C. 25 Misused Words that Make You Sound (Or Look) Dumb. http://www.yourdictionary.com/slideshow/25-misused-words-make-you-sound-dumb.html. Accessed: 2018-01-10.

[93] LoveToKnow, C. Examples of Homonyms. http://examples.yourdictionary.com/examples-of-homonyms.html. Accessed: 2018-01-10.

[94] LoveToKnow, Corp. Examples of Tautology. http://examples.yourdictionary.com/examples-of-tautology.html. Accessed: 2018-01-10.

[95] Ltd., T. O. R. G. Top 10 tips for writing a dissertation methodology. https://www.oxbridgeessays.com/blog/top-10-tips-writing-dissertation-methodology/. Accessed: 2017-09-30.

[96] Margino Apps. Synonym & Antonym Dictionary. https://play.google.com/store/apps/details?id=com.marginoapps.synonymantonym. Accessed: 2017-09-30.

[97] Markus Schmidt, Anna Gerdt, P. S. Google Assistant: Neue Funktionen ... aber vorerst nur in den USA! http://www.computerbild.de/artikel/cb-News-App-Check-Google-Now-Google-Assistant-16367107.html. Accessed: 2018-01-05.

[98] Massachusetts Institute of Technology. Setting Up App Inventor. http://appinventor.mit.edu/explore/ai2/setup.html. Accessed: 2018-02-10.

[99] McGraw, I., Prabhavalkar, R., Alvarez, R., Gonzalez Arenas, M., Rao, K., Rybach, D., Alsharif, O., Sak, H., Gruenstein, A., Beaufays, F., and Parada, C. Personalized Speech recognition on mobile devices. *ArXiv e-prints* (Mar. 2016).

[100] Microsoft Inc. Advanced English Dictionary and Thesaurus. https://www.microsoft.com/en-us/store/p/advanced-english-dictionary-and-thesaurus/9wzdncrdp7tp. Accessed: 2018-02-10.

[101] Microsoft Inc. Cortana, Ihre persönliche digitale Assistentin. https://www.microsoft.com/de-de/windows/cortana. Accessed: 2018-02-10.

[102] Microsoft Inc. Introducing Harman Kardon INVOKE with Cortana by Microsoft. https://www.microsoft.com/en-us/cortana/devices/invoke. Accessed: 2018-02-10.

[103] Microsoft Inc. Microsoft cortana. https://www.microsoft.com/en-us/windows/cortana. Accessed:.

[104] Microsoft Inc. Microsoft Übersetzer. https://play.google.com/store/apps/details?id=com.microsoft.translator. Accessed: 2017-09-30.

[105] Microsoft Inc. Was ist Cortana? https://support.microsoft.com/de-de/help/17214/windows-10-what-is. Accessed: 2018-02-10.

[106] Microsoft Inc. What is cortana? https://support.microsoft.com/en-us/help/17214/windows-10-what-is. Accessed:.

[107] MNG Mobile. English Idioms & Phrases. https://play.google.com/store/apps/details?id=mng.com.idiomandphrasal. Accessed: 2017-09-30.

[108] MS apps. All English Idioms & Phrases. https://play.google.com/store/apps/details?id=com.app.englishidioms. Accessed: 2017-09-30.

[109] Muntenescu, F. Android Architecture Patterns Part 1: Model-View-Controller. https://medium.com/upday-devs/android-architecture-patterns-part-1-model-view-controller-3baecef5f2b6. Accessed: 2018-04-14.

[110] Muntenescu, F. Android Architecture Patterns Part 2: Model-View-Presenter. https://medium.com/upday-devs/android-architecture-patterns-part-2-model-view-presenter-8a6faaae14a5. Accessed: 2018-04-14.

[111] Muntenescu, F. Android Architecture Patterns Part 3: Model-View-ViewModel. https://medium.com/upday-devs/android-architecture-patterns-part-3-model-view-viewmodel-e7eeee76b73b. Accessed: 2018-04-14.

[112] Navneet. Android Speech to Text Tutorial. https://www.androidtutorialpoint.com/material-design/android-speech-text-tutorial/. Accessed: 2017-12-05.

[113] Passuello, L. Top 3 Reasons to Improve Your Vocabulary. https://litemind.com/top-3-reasons-to-improve-your-vocabulary/. Accessed: 2018-03-10.

[114] Patil, R. Pros and Cons of Cross-Platform Mobile App Development. https://www.infoq.com/articles/mobile-cross-platform-app-development. Accessed: 2018-02-10.

[115] PayScale Inc. Salary Data & Career Research Center (Germany). https://www.payscale.com/research/DE/... Accessed: 2017-09-30.

[116] PayScale Inc. Salary Data & Career Research Center (Spain). https://www.payscale.com/research/ES/... Accessed: 2017-09-30.

[117] Pöhland, J. False Friends in English and German Vocabulary. https://www.englisch-hilfen.de/en/words/false_friends.htm. Accessed: 2018-01-10.

[118] próceres, L.

[119] programcreek. Java Code Examples for com.android.volley.RequestQueue. https://www.programcreek.com/java-api-examples/index.php?api=com.android.volley.RequestQueue. Accessed: 2018-02-10.

[120] Radigan, D. Kanban. https://de.atlassian.com/agile/kanban. Accessed: 2018-01-30.

[121] Saldanha, C. Tutorial: Using Wordnik and Big Thesaurus API. http://cjds.github.io/tutorial/nlp/2014/04/04/Knicker-Big-Huge-Thesaurus-tutorial/. Accessed: 2018-02-10.

[122] Samsung Group. A smarter way to use your phone. http://www.samsung.com/global/galaxy/apps/bixby/. Accessed: 2018-02-10.

[123] Samsung Group. A smarter way to use your phone. http://bixby.samsung.com/. Accessed: 2018-02-10.

[124] Samsung Group. HOW CAN WE HELP YOU? https://help.content.samsung.com/csweb/faq/searchFaq.do. Accessed: 2018-02-10.

[125] Samsung Group. Intelligence. http://www.samsung.com/uk/smartphones/galaxy-s8/intelligence/. Accessed: 2018-02-10.

[126] Samsung Group. Intelligenz. http://www.samsung.com/de/smartphones/galaxy-s8/intelligence/. Accessed: 2018-02-10.

[127] SaveOnEnergy.com. Estimating Electricity Usage. https://www.saveonenergy.com/energy-consumption/. Accessed: 2018-04-14.

[128] Sekhar, C. Any Dictionary Definition API for Java? https://stackoverflow.com/questions/11714477/any-dictionary-definition-api-for-java. Accessed: 2018-02-10.

[129] Siepermann, D. M. Agile Softwareentwicklung. https://wirtschaftslexikon.gabler.de/definition/agile-softwareentwicklung-53460. Accessed: 2017-09-30.

[130] Soanes, C. What's the difference between 'will' and 'shall'? https://blog.oxforddictionaries.com/2013/09/25/will-versus-shall/. Accessed: 2017-10-10.

[131] Softissimo Inc. Context Reverso. https://play.google.com/store/apps/details?id=com.softissimo.reverso.context&hl=en. Accessed: 2017-09-30.

[132] Softissimo Inc. Context Reverso. http://context.reverso.net/translation/mobile-app. Accessed: 2017-09-30.

[133] Spier, A. Agile Softwareentwicklung gestern, heute und morgen. https://jaxenter.de/agile-softwareentwicklung-gestern-heute-und-morgen-2719. Accessed: 2017-09-30.

[134] Suttle, R. Definition of Business Objectives & Goals. http://smallbusiness.chron.com/definition-business-objectives-goals-24983.html. Accessed: 2017-12-14.

[135] Tamada, R. Android Speech To Text Tutorial. https://www.androidhive.info/2014/07/android-speech-to-text-tutorial/. Accessed: 2017-12-05.

[136] ter Horst, J. Thesaurus API. https://www.programmableweb.com/api/thesaurus. Accessed: 2018-02-10.

[137] TextRazor Ltd. Languages. https://www.textrazor.com/languages. Accessed: 2018-02-10.

[138] TextRazor Ltd. TextRazor. https://www.textrazor.com/. Accessed: 2018-02-10.

[139] TextRazor Ltd. TextRazor. https://www.textrazor.com/tutorials. Accessed: 2018-02-10.

[140] TextRazor Ltd. TextRazor Java Reference. https://www.textrazor.com/docs/java. Accessed: 2018-02-10.

[141] Thesaurus. Java. http://thesaurus.altervista.org/testjava. Accessed: 2018-02-10.

[142] Thesaurus. Thesaurus. http://thesaurus.altervista.org/. Accessed: 2018-02-10.

[143] Travis Skare. Get notifications from Google Now in Chrome. https://chrome.googleblog.com/2014/02/get-notifications-from-google-now-in.html. Accessed: 2018-02-11.

[144] U.S. Environmental Protection Agency. Greenhouse Gas Equivalencies Calculator. https://www.epa.gov/energy/greenhouse-gas-equivalencies-calculator. Accessed: 2018-04-14.

[145] user: French Man. The French Man Who Went To Malta. https://www.youtube.com/watch?v=PFkzSfRFiMU. Accessed: 2018-02-13.

[146] user: Fry J. Apocaloso. The Italian Man Who Went To Malta (ORIGINAL ANIMATED VERSION)- 2009. https://www.youtube.com/watch?v=YjXGywPzkw0. Accessed: 2018-02-13.

[147] user: Keira. Notes from Spain: Any funny story about language misunderstandings when learning Spanish? (or English). http://www.notesfromspain.com/forums/showthread.php?t=7223.

[148] user: Teacher Frank. The Italian man who went to america. https://www.youtube.com/watch?v=yvGjqDLyCos. Accessed: 2018-02-13.

[149] VersionOne Inc. The 10th Annual State of Agile Report. https://www.versionone.com/pdf/VersionOne-10th-Annual-State-of-Agile-Report.pdf. Accessed: 2017-09-30.

[150] Wagner, A. Two Major Challenges with Speech-Recognition Technology. https://uxmag.com/articles/two-major-challenges-with-speech-recognition-technology. Accessed: 2017-12-05.

[151] Watson, J. Big Huge Thesaurus, About. https://words.bighugelabs.com/about.php. Accessed: 2018-02-10.

[152] Watson, J. Big Huge Thesaurus, API. https://words.bighugelabs.com/api.php. Accessed: 2018-02-10.

[153] Wheatcraft, L. Using the correct terms - Shall, Will, Should. http://reqexperts.com/blog/2012/10/using-the-correct-terms-shall-will-should/. Accessed: 2017-10-10.

[154] Wholesale Solar. How Much Power Do Your Appliances Use? https://www.wholesalesolar.com/solar-information/how-to-save-energy/power-table. Accessed: 2018-04-14.

[155] Wikimedia Foundation. Wikimedia Downloads. https://dumps.wikimedia.org/backup-index.html. Accessed: 2018-02-10.

[156] Wintertree Software Inc. Wintertree Thesaurus Engine Java SDK. https://www.wintertree-software.com/dev/thesdb/java/technical.html. Accessed: 2018-02-10.

[157] WireframeSketcher.com. Download WireframeSketcher. https://wireframesketcher.com/download.html. Accessed: 2018-02-01.

[158] Wu, D. The Italian Who Came To America. https://www.flickr.com/photos/fractalife/3032665066. Accessed: 2018-02-13.

[159] Yandex. Yandex.Translate - offline translator & dictionary. https://play.google.com/store/apps/details?id=ru.yandex.translate. Accessed: 2017-09-30.

[160] Yevgen Chebotar, Iryna Gurevych, C. M. M. C. M. L. Q. T. Z. DKPro JWKTL. https://dkpro.github.io/dkpro-jwktl/. Accessed: 2018-02-10.

[161] Yevgen Chebotar, Iryna Gurevych, C. M. M. C. M. L. Q. T. Z. JWKTL. https://mvnrepository.com/artifact/de.tudarmstadt.ukp.jwktl/jwktl/0.17.2. Accessed: 2018-02-10.

[162] Yevgen Chebotar, Iryna Gurevych, C. M. M. C. M. L. Q. T. Z. JWKTL. https://www.ukp.tu-darmstadt.de/software/jwktl/. Accessed: 2018-02-10.

List of Figures

List of Tables

Appendices

Voice assistant services (more detailed)

Voice assistant services are intelligent assistants which the user controls with his voice. Like mentioned in chapter 3.2 Voice assistant services I took a look on the five popular services by big companies and will describe them more detailed in the following paragraphs. These services are Siri by the Apple Inc., Amazon Echo by the Amazon Inc., Cortana by Microsoft, the Google Assistant and Samsungs Bixby.

Siri

Siri is the voice service for Apple products. A goal of Siri is to make the usage of the products easier, faster and it is always available for the user whatever Apple product he is using. Siri is available on the smart-phone (iPhone), touch-pad (iPad), notebook (Mac), smartwatch (Apple Watch), television equipment (Apple TV) and speaker (HomePod). The user can start Siri by saying "Hey Siri" and than he adds his spoken request. This could be a question about anything he wants to know or e.g. the call of a specific service like the weather report, that Siri should take a photo, send some messages or start phone calls. Siri also manages tasks, reminders, timer and others. This makes Siri to a personal assistant or secretary. A dictating function not only for messages works as well. In case that Siri understand the user wrong he can tap and correct the voice input by typing. Among this Siri has access to other apps which it can open for the user.
The functional range depends on the country but in addition to the mentioned tasks Siri even has general features for sport, a Twitter integration, information, reviews, reservations for restaurant, information, reviews, show time of movies, a dictionary and a calculation feature. [15] [16] [14] [12]

Amazon Echo

Amazon Echo is a voice assistant service that is known as Alexa as well. There are two variations of Amazon Echo (more variations in the USA): Amazon Echo itself and the smaller version Amazon Echo Dot. Both are speakers which are listing to the voice input of users in different languages. That works even from across a room and in a loud room where e.g. music is playing. The voice assistant services of both variations is the same only the hardware is different.
Amazon Echo supports his users by a hands-free usage and with a lot of different functions. In one hand it can play music, read audiobooks and news like a radio. In doing so all information e.g. traffic and weather report are more specific for the user and belong to his location or request. The user can ask about local stores, schedules and scores of sport teams and others. Furthermore the user of Amazon Echo is able to call or send messages with it as well and it has an intelligent alarm clock that his user can use e.g. for sleeping or cooking.

On the other hand Amazon Echo provide a hand-free control for smart home devices. Among others it is possible to control light, switches, television, fans, thermostats, locks with Amazon Echo.
Amazon Echo is learning continuous to get smarter. It is always automatically up to date over cloud where it gets new features/functions and skills. [8] [7] [6]

Cortana

The voice service Cortana is a digital assistant from Microsoft. It saves information about his user to improve it's working and become more useful for him. Like the other voice services it is dealing with basic tasks like answering questions, telling informations about the weather and latest news. It creates lists and reminder which bases on time, location or situations, controls the music, finds restaurants and is able to manage smart home devices like dimming your lights or adjusting the thermostat.
The user can have Cortana not only on his phone, even when it is on his notebook the user is able to call every Skype-enabled device like mobile phones and landlines by saying it to Cortana. To start Cortana it is enough to say "Hey Cortana" and the service works with whole sentences that the user can type or say. An example at microsoft.com [106] is "Remind me to congratulate Tanya the next time she calls.". This is an example for the reminder based on the situation of an incoming call by a specific person.
Microsoft gives a look ahead and advertise a next feature of Cortana. It should be able to read emails loud for it's user [102].[102] [101] [105] [103] [106]

Google Assistant

Google Now is an app for android devices, iPhones and iPads. This app is assessable by voice and typing to inform the user about the weather, news and interesting stories like live scores of sports team, public transport, traffic conditions and alternative routes depending on the time and location, nearby attractions, restaurant reservations and flights. As combination of this Google Now has a reminder for upcoming appointments and e.g. it tells his user when he has to leave depending on his location and the traffic conditions.
Furthermore Google Now is able to send text messages, play music and it orders the apps by suggestions which app the user is looking for to support him. The app provides a fast search form the home screen or by saying "Ok Google" and telling the task for the app. [56] [59] [57] [58] [143]
The Google Assistant is a kind of combination of the Google search engine and Google Now to fast up the support of it's user and give him individual results more quickly. It adapt itself to the interests of it's user.
Among others the Google Assistant is available in Google products like Google's messaging app Allo and Google Home. Google Home is a hardware speaker like Amazon Echo. [55]
Like Google Now the Google Assistant connects all Google apps of an user. It works by selecting information about the usage to improve itself for the personal needs of it's user. That means the Google Assistant remembers previous questions and is able to answer

following questions by knowing the context. The user can ask for movies in the cinemas next to him and add that he wants to take his children with him. Than the Google Assistant offers him age-appropriate films for the children. The example continues by the task ”Show me the trailer”. Without the need to repeat the name of the movie, the user can asks ”Who is the producer?” and than ”How old is he?”. The Google Assistant answers this questions by knowing the context and the user doesn’t need to repeat the name of the movie again or the name of the producer in his questions. [40] [39] [97]

Bixby

Bixby is an intelligent, personalized voice interface for Samsung phones. It is introduced by the Samsung Group for the Galaxy S8 and later models. The user can control Bixby hands-free by talking in English, Korean or Mandarin Chinese. Other ways to control are touching and typing.
To support the user the best, Bixby is studying his behaviours and tries to fit the app content to the users routine. It has access to other apps and knows incoming messages, calls, and upcoming events. In addition Bixbys reminder function bases on time, location and situation. E.g. this voice service reminds his user when he comes to the office to call someone back or to buy milk when he is next to the supermarket. Compared to others Bixby can include a video to resume, Contana only facilitate images and Amazon Echo has no visual representation (only at the app).
Get Bixby to remind you to water the plants when you get home or to call someone back when you get back to the office. You can also have the reminders show on the AOD screen, Bixby Home, or the edge screen. Another difference to the other voice services is the photo feature. Bixby promote itself with the offered help by the camera. It can translate or get information about an object on the photo e.g. recognition what the user sees and online shopping after scanning a bar code of a product. [124] [122][123] [125] [126]

Summary of voice assistant services

All analysed voice assistant services can understand at least the English spoken language. Most of them are not only controllable by voice. Tapping and texting are working as well. The use of the voice assistant service only with the voice without anything else makes a faster and hand-free usage accessible. Therefore the alternative way to start the services only by saying a code word is useful. But than the service always have to listen and analyse everything for this code word.
It looks like the basic functions of a voice assistant service are the calendar with reminder, music player, weather and news report, calling and messaging. I think one of the most important functions thereby is to answer the questions of the users. Therefore they use different search engines for example Google (Google Assistant) and Bing (Amazon Echo) [35].
To be a good assistant the voice assistant services study the behaviours and routines of their user and work with informations like the location and context knowledge between

different apps. Thereby they can give the user specific information at the time when he needs them even when he didn't ask for them. An example is the worker which has an appointment after work. An assistant service could give him a hint about the traffic conditions to leave work earlier because of his location, a high traffic and the time that he will need to be punctual at the other place [143].
Nevertheless the services are not only connecting informations from a handful apps. The functions of some voice assistant services are expandable by more applications or devices of other supplier and devices. The compared voice assistant services are all compatible with different devices and can synchronize between applications. That makes them good for e.g. supporting a smart home. Indeed some of the voice assistant service devices are made to stay at home and can't be taken everywhere e.g. Amazon Echo and Google Home.

The Italian Who Came To America

The spelling mistakes in the following text ”The Italian Who Came To America” are necessary for a better understanding of the problem. The text is taken literally from the source [158].

One day Ima go to No Fock, Virginia to a Bigga Hotel. I go down to eat soma breakfast.

I tell the waitress I wanna two piss toast. She branga me only onea piss. I tell her I wanna two piss, she say go to the toilet, I say you no understand. I wanna two piss on my plate. She say you better no piss on the plate you Sonna Ma Bitch. I don't even know the lady and she call me Sonna Ma Bitch.

Later I go to eat some lunch at Tyler Restaurant, the waittress, Virginia, bringa me a spoon, ana knife, but no fock. I tell her I wanna fock, Virginia, She tellsa me everybody wanna fock. I tell her you no understand, I wanna fock on the table. She say no fock Virginia on the table. You sonna Ma Bitch. I don't eve know the lady and she calls me Sonna Ma Bitch and get the hell out of No Fock Virginia.

So I go back to my room inna hotel, an there's no sheet on my bed. I calla the manager ana tell him I wanna sheet. He tells me to go to the toilet. so I say you no understand, I wanna sheet on the bed. He say you better not sheet onna bed, you Sonna Ma Bitch. I don't even know the man and he call me Sonna Ma Bitch.

I go to check out and the man at the desk, he say ”Peace to you”. I say piss onna you, too, you Sonna Ma Bitch. I go back to Italy!!!

The commonest English words

The commonest English Words in alphabetical order. [75] [80] [87] [5]

- a
- about
- after
- again
- all
- also
- am
- an
- and
- any
- are
- as
- at
- back
- be
- because
- been
- before
- being
- but
- by
- call
- can
- come
- could
- day
- did
- do
- down
- each
- even
- find
- first
- for
- from
- get
- give
- go
- good
- had
- has
- have
- he
- her
- him
- his
- how
- I
- if
- in
- into
- is
- it
- its
- just
- know
- like
- little
- long
- look
- made
- make
- man
- many
- may
- me
- might
- more
- most
- Mr
- Mrs
- much
- must
- my
- never
- new
- no
- not
- now
- number
- of
- oil
- on
- one
- only
- or
- other
- our
- out
- over
- own
- part
- people
- said
- say
- see
- she
- should
- so
- some
- such
- take
- than
- that
- the
- their
- them
- then
- there
- these
- they
- think
- this
- time
- to
- two
- up
- upon
- us
- use
- very
- want
- was
- water
- way
- we
- well
- were
- what
- when
- which
- who
- will
- with
- word
- work
- would
- write
- year
- you
- your

Code

The following listings show snippets of Code.

```
//Receiving speech input
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    switch (requestCode) {
      case REQ_CODE_SPEECH_INPUT: {
         if (resultCode == Activity.RESULT_OK && null != data) {
           String recordResult = data
           .getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS).get(0);
           if(voiceInputView.getText().toString().equals(
             getString(R.string.editTxt_voiceInputView))){
                 voiceInputView.setText("");
              } else {
                 voiceInputView.append("\n");
              }
              String newInput = filterSearchType(recordResult);
              voiceInputView.setText(voiceInputView.getText() + newInput);
              listAdapter.notifyDataSetChanged();
              startSearch();
           }
           break;
       }
    }
}
```

Listing 9: Code: get voice input

```
public enum SearchType {
   contextRazor("Context"),
    synonym("Synonym"), antonym("Antonym"),
    equivalent("Equivalent"),
    related_word("Related word"),
    rhyme("Rhyme"),
    etymologically_related_term("Etymologically related term"),
    form("Form"),
    hypernym("Hypernym"),
    inflected_form("Inflected form"),
    primary("Primary"),
    hyponym("Hyponym"),
    variant("Variant"),
    verb_stem("Verb stem"),
    verb_form("Verb form"),
    same_context("Same context");

    private String searchType;
    private SearchType(final String word){
        this.searchType = word;
    }

    public String getSearchType() {
        return searchType;
    }
    public static boolean contains(String test) {
```

```java
       for (SearchType st : SearchType.values()) {
          if (st.name().equals(test)) {
             return true;
          }
       }
       return false;
    }
}
```

Listing 10: Code: the enum SearchType

```java
private void getWordnik(String word, Knicker.RelationshipType type) {
    List<Related> list;
   try{
      list = WordApi.related(word, true, EnumSet.of(type), 10);
      for (Related r : list) {
         for (String s : r.getWords()) {
            addResult(s);
         }
      }
   } catch (KnickerException e) {
        e.printStackTrace();
   }
}
```

Listing 11: Code: getting results by Wordnik

```java
private void getThesaurusSynonym(String word) {
    String words;
   try {
      URL url = new URL("http://words.bighugelabs.com/api/"
         + "1/9a0d1e46117e2bdb3bf6e1a1568faa3e/" + word + "/");
      InputStreamReader isr = new InputStreamReader(url.openStream());
      BufferedReader br = new BufferedReader(isr);
      try {
         while ((words = br.readLine()) != null) { // while loop to read all
         for (String search:strValues) {
            if (search.equals(words)){
               break;
               }
            }
            addResult(words);
         }
      } finally {
         br.close();
         isr.close();
      }
    } catch (NetworkOnMainThreadException | IOException e) {
      e.printStackTrace();
    }
}
```

Listing 12: Code: getting synonyms by Big Huge Thesaurus

```
private void getTextRazorStem(String words) {
    AnalyzedText response = setUpTextRazor(words);
    if (response != null && response.getResponse() != null) {
      if (response.getResponse().getSentences() != null) {
        for (Sentence sentence : response.getResponse().getSentences()) {
          for (Word word : sentence.getWords()) {
            addResult(word.getStem());
          }
        }
      }
    }
}

public void getTextRazorContext(String words) {
    AnalyzedText response = setUpTextRazor(words);
    if (response != null && response.isOk()
         && response.getResponse() != null) {
      for (Custom custom : response.getResponse().getCustomAnnotations()) {
        // get content
        for (Custom.BoundVariable variable : custom.getContents()) {
          if (null != variable.getEntityValue()) {
            for (Entity entity : variable.getEntityValue()) {
              addResult(entity.getEntityId());
              getWordnik(entity.getEntityId(),
                  Knicker.RelationshipType.hypernym);//special: Wordnik
            }
          }
        }
      }
      // get topics
      if (response.getResponse().getTopics() != null) {
        for (Topic topic : response.getResponse().getTopics()) {
          addResult(topic.getLabel());
        }
      }
      // get entailments
      if (response.getResponse().getSentences() != null) {
        for (Sentence sentence : response.getResponse().getSentences()) {
                for (Word word : sentence.getWords()) {
                    for (Entailment entailment : word.getEntailments()) {
                        addResult(entailment.getEntailedWords().get(0));
                    }
                }
            }
        }
    }
}
```

Listing 13: Code: setting up and getting results by TextRazor

```
public class WordTextRazor {
    private static TextRazor client;
    private final HashMap<String, Integer> resultMap = new HashMap<>();
    public WordTextRazor() {
        setAnalysisRules();
    }

    private void setAnalysisRules() {
        // Sample request, showcasing a couple of TextRazor features
        client = new TextRazor( //my API key
      "bf9c9a22d9c7ea8a4c66e1068b773af28f62d248fd982699a3000afc");

        client.addExtractor("words");
        client.addExtractor("entities");
        client.addExtractor("topics");

        client.setClassifiers(Collections.singletonList(
         "textrazor_newscodes"));

        client.addExtractor("entailments");//
        client.addExtractor("senses");
        client.addExtractor("phrases");//
        client.addExtractor("dependency-trees");
        client.addExtractor("relations");//

        client.addExtractor("entity_companies");

        client.setEnrichmentQueries(Arrays.asList("fbase:/location/location" +
         "/geolocation>/location/geocode/latitude",
            "fbase:/location/location/geolocation>/location/" +
         "geocode/longitude"));

        String rules = "entity_companies(CompanyEntity) :- " +
         "entity_type(CompanyEntity, ’Company’).";
        client.setRules(rules);
    }

    public AnalyzedText setUpTextRazor(String words) {
        setAnalysisRules();
        AnalyzedText result = null;
        if (client == null || words == null || words.length() < 1) {
            return result;
        }
        try {
            result = client.analyze(words);
        } catch (RuntimeException | NetworkException | AnalysisException e) {
            e.printStackTrace();
            return null;
        }
        return result;
    }

    private void addResult(String str) {
        if (!str.isEmpty() || str.equals(" ")) {
            if (resultMap.get(str) != null) {
                int i = resultMap.get(str) + 1;
                resultMap.put(str, i);
            } else {
                resultMap.put(str, 1);
            }
        }
    }
   ... // getter methods
}
```

Listing 14: Code: class to use the TextRazor

Declaration of Authorship

I certify that the work presented here is, to the best of my knowledge and belief, original and the result of my own investigations, except as acknowledged, and has not been submitted, either in part or whole, for a degree at this or any other university.

Barcelona, April 18, 2018

A.-K. Hannemann