Technische Universität Berlin

DOCUMENT BUILD DATE: 14th May 2018
DOCUMENT STATUS: Semi-Final Draft Part 1

Development and Evaluation of a Service Bot for the e-Government Sector

Bachelor Thesis
am Fachgebiet Agententechnologien in betrieblichen Anwendungen und der Telekommunikation (AOT)
Prof. Dr. Dr. h.c. Şahin Albayrak
Fakultät IV Elektrotechnik und Informatik
Technische Universität Berlin

vorgelegt von
Mohamed Megahed

Betreuer: **Dr. Andreas Lommatzsch**
Gutachter: Prof. Dr. Dr. h.c. Şahin Albayrak
Prof. Dr. Odej Kao

Matrikelnummer: 342655

Declaration of Authorship

I, MOHAMED MEGAHED, hereby declare that the thesis submitted is my own work, completed without any unpermitted external help. Only the sources and resources listed were used.

The independent and unaided completion of the thesis is affirmed by affidavit:

Berlin, 14th May 2018

Erklärung der Urheberschaft

Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher in gleicher oder ähnlicher Form in keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Ort, Datum
Berlin, 14th May 2018

Unterschrift

Abstract

Though not a recent phenomenon, chatbots and voice assistants are increasingly gaining unprecedented attention as a successor for mobile and web applications. While still emerging with no defined standards or set protocols, with a hype on the rise, tensions between industry giants with products like Amazon's Alexa, Apple's Siri or the Google Assistant unveil new examples for providing an enriched user experience for consumers and businesses. The surrounding ecosystem also plays a major role in spreading the platforms available while exploring new horizons with alternative approaches and business models. Today, voice assistance are already present around indoor spaces, in the car or in our pockets on the go. They live inside smartphones, on fancy TV sets or even as stand-alone devices. Yet, they are still a new terrain to discover and great potential to unleash in numerous contexts.

One such scenarios involves the public sector. In this work, we create an Alexa Skill taking a critical approach to the artefacts provided by Amazon in combination with respective services and artefacts to develop a voice assistant for the local city council giving a parallel experience to its current VIRTUAL CITIZEN ASSISTANT - *Virtueller Bürgerassistent* available on The Berlin City Portal website. We touch on the technical challenges and possibilities in implementing a Software as a Service (SaaS) system for e-Government inquiries and analyse its usability as well as its effectiveness as an additional option to traditional information lookup. Examine the goals we define through a survey, we conclude a high potential to adopt the Skill once published.

With a critical approach to our use case given the APIs and SDKs available at the time, we are able to make the Skill simulate the telephone hotline D115 for many simple berlin services and provide a proof-of-concept that more complex issues can be handled through simple content management with no need for redeployment. we also report on the opportunities, challenges and limitations faced in the development process.

With our study of the current state of voice assistants and service bots on the market, we give recommendation of best practices to develop a voice assistant.

Zusammenfassung

Chatbots und Sprachassistenten nehmen ein wachsendes Interesse in den letzten zwei Jahren. Im Gegensatz zum öffentlichen Email-Protokoll, konkurrieren Groß- und Kleinunternehmen auf eine universale Systemausprägung und diversifizieren mithilfe von künstlicher Intelligenz den Rahmen für den persönlichen Assistenten der Zukunft. Ob es durch Apples Siri, Cortana von Microsoft oder den Google Sprachassistenten hervorgeht, zeigen sich zahlreiche Beispiele, wie die Nutzererfahrung auf privaten oder Geschäftsebene weitgehend bereichert werden kann. Der steigende Nutzen von Sprachassistenten zuhause, im Büro und unterwegs erlaubt viele neue Erkenntnisse und Erfahrungen, die dem Marktumfeld zu verdanken sind.

Auch auf staatlicher Ebene wird dieser Nutzen dank einer wachsenden Rolle von E-Government in der Gesellschaft vertreten. Der Beitrag von Sprachassistenten hat einen Einfluss auf dem Onlinedienstindex eines Landes und trägt zu einer wichtigen Modernisierung im öffentlichen Wesen bei.

In der folgenden Recherche untersuchen wir Alexa von Amazon im Zusammenhang mit anderen verfügbaren Plattformen und Artefakten. Wir erweitern die Fähigkeiten des Sprachassistenten durch den Webauftritt des berliner Stadtportals (BerlinOnline) sowohl für die vielfältigen und mehrsprachigen Einwohner der Hauptstadt als auch deren Senatsverwaltung. Als Grundlage wird auf die Funktionalität des VIRUTLEEN BÜRGERASSISTENTEN Berlins zurückgegriffen. Anhand dieses Anwendungsfalls werden die prinzipiellen Möglichkeiten und Herausforderungen bei der Implementierung eines Sprachassistenten für die Stadtverwaltung aus technischer und organisatorischer Sicht analysiert. Zusätzlich wird die Usability des Systems als Ergänzung für einen Abfragedienst in Erwägung gezogen. Lösungen aus dem Chatbot der Stadt Wien sowie der Sprachassistent von Georgien, USA werden zum Vergleich betrachtet.

Produkt dieser Arbeit ist ein Alexa Skill, welches die Basis des vorhandenen Chatbots nutzt, um Anfragen an der Berliner verwaltung zu beantworten sowie eine Umfrage zur vorläufigen Erfolgsbemessung des Produkts. Der Skill simuliert die Telefonhotline D115 für einfache Anfragen und zeigt, wie komplexere Themen auch nur durch Content-Management gehandhabt werden können. Die Umfrage unterstützt die Softwareentwicklung und zeigt, dass die Nachfrage nach einem Sprachassistenten in der Verwaltung einen hohen Erfolgspotential bei einer Inbetriebnahme verspricht.

Acknowledgements

[*Supervisor*] BETREUERNOTIZEN SIND IM FLIESSTEXT MIT \SN{TEXT}]

⌈ possibility to make inline notes ⌋
[CITATION] when a citation is missing

✍ **TO-DO**: \todo{text}

[TO CITE]

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With over a third of the world's population projected to own a smartphone in 2018 [40] and a substantial fraction thereof using smarthome gadgets and appliances on a daily and even hourly basis, the role of Artificial Intelligence (AI) has become more interesting than ever in many disciplines including but not limited to productivity and entertainment. Many technologies we take for granted today, such as dictation and word prediction, recommender systems or other digital analytics depend on Machine Learning (ML) and Natural Language Processing (NLP) techniques that were only made possible thanks to the high computational power shipped in most devices gradually overtaking the consumer market. This transition also facilitated the introduction of a new form of interaction through conversation with the hardware, paving the way to an aspiration modern societies have been striving globally [22]. Whether in blockbuster 60s drama as seen in "Breakfast at Tiffany's" (1961) or in Sci-Fi romance in the movie "Her" (2013), an obsession with voice technologies is featured throughout from an answering machine on tape to a fully personalized but mass-produced voice-based operating system to even become a protagonist.

Conversational bots were already prevalent since the 80s in the form of Question Answering Systems based on query programming languages like PROLOG and SQL. ELIZA, considered as the world's first chatbot and though quite superficial as an NLP-based programme for psychoanalysis, already at its early stages demonstrated how humans can become emotionally attached to machines, transcending over the anomaly of making conversation *not* with a human [48]. Today, combining ML with retrieval-based approaches allows a more advanced interaction with the system and yields smarter and more personalized conversations between man and machine. Consequently, it is no longer a surprise that chatbots acquire social skills to make Xiaoice, the empathetic bot from China, possibly a new kind of friend made of silicon revoking the fiction element from "Her".

So far, voice assistants represent an additional layer of abstraction from software beyond the graphical user interface (GUI) and are hence the closest we have come towards human communication. They deconstruct another barrier between the user and the hardware as voice communication generally does not require profound computer literacy and the conversational models rely on our inherent ways of expression. As they simulate the human aspect and imitate its behaviour for instance with small-talk abilities [39], voice assistants are regarded as a convenience for daily tasks and are on their way to becoming a de-facto replacement to sophisticated actions we perform on the screen. In fact voice searches now compose a large market segment with over a billion voice searches per month [1] and are predicted to make up about "1 Billion Voice Assistant enabled devices in circulation by the end of 2018" [1] and 30% [15] [42] of all searches by 2020 with currently highest rates coming from the youngest generations based on a survey [49] by Global Web Index. According to the Alpine.ai 2017 Voice Labs report there were 33 million voice-first devices in circulation in 2017 worldwide [2] with tremendous shifts in number of units sold between 2015 and 2017. This and more statistics hint at a revolutionary change towards our use as much as the introduction of the GUI and mouse were to the Command Line Interface (CLI). Increasingly, we recognize voice as a *new* user interface, also known as Voice User Interface (VUI), and analyse good practices and design guidelines for it.

Since sound and voice are primal stimuli in the human brain [18], using them becomes more instinctive, making us in turn process our ideas starting with an "inner voice" that translates easiest into words when we speak it before routing or "funneling" it to actions [7]. Arguably, on a market scale, this gives voice assistants today a competitive innovation advantage (CIA) for they are hence more accessible to further demographic groups with a growing wider acceptance.

Meanwhile, statistics from BusinessInsider [25] show that time spent on messaging applications (apps) already surpassed average uptime on social media, which indicates how the former is more desirable as a communication format on mobile platforms and how having a conversation is an easier way to interact with a device instead of downloading an app for every task. Further, the speech-to-text/text-to-speech domain has become more powerful with steadily increasing processing power, an effect of Moore's law we only get to understand lately in addition to the gradual lessening of the dominance of Chomskyan theories of linguistics (e.g. transformational grammar) [9]. With an integration in most modern operating systems, speaking to a device has become no longer a absurd novelty. Inadvertently, the Google voice assistant built into its Android OS currently with the highest usage shares among computer and smartphones [11], supports understanding of multiple languages in the same sentence for multilingual interaction. Moreover, the Echo Dot recorded the peak for best-sellers

for 2017 on Amazon with unprecedented numbers showing a high customer retention and satisfaction rate [34]

As we constantly challenge our expectations towards technology, our imagination makes us question the ability of AI to make a machine able to react to everything we say, which can be not too far-fetched in a near future. Although we are still far from this step, at least for the consumer level, we dedicate a lot of effort to make it happen with examples like IBM's Watson or other Uses of big data analytics. We can probably conclude though, that as long as we still do not exactly understand human intelligence in detail yet, it is hard to fathom AI as a holistic field. As such, it is therefore more realistic to consider the current works in the field as *intelligence amplification* [7] empowering human take better decisions beyond their normal brain processing power and not overtaking human intelligence as some might claim.

1.1 Motivation

With the aforementioned, we try to think how voice assistants could come handy and why we want to invest in such technology, juxtaposing it to available alternatives. If we consider human workforce (i.e. customer service agents) on one hand, it is commonplace that these are more expensive, less available due to restrictive working hours and not always aware of the full circumstances related to an issue they are supposed to fix or a question they are to answer. Sometimes knowing more about a person could almost become a dangerous tool since it gives room to manipulate them. A client in a shop for instance can have their decision influenced by the seller and eventually get tricked into buying a product based on wrong advice. Although there is practically little the client can normally do to circumvent misinformation if that seller is replaced by an algorithm, having an automated system like a voice assistant step in gives at least a more neutral impression since it are not directly expected to act with malicious intentions like a vendor who abuses the client's trust.

Since the point of availing voice assistant is to act in a person's interest, we also want an information system not to confuse us or to limit our cognitive abilities. Besides, we can at least ensure in a system design that a voice assistant will not become moody and intentionally want to make our lives harder for this reason as opposed to a human. And so, although a voice assistant or a chatbot may not potentially answer every question we throw at it, we want to at least presume that it would give us no information instead of partial truths or lies while keeping a certain level of neutrality and "decency" in terms of the wording. This is why most credible companies elaborate explicitly in their terms, conditions and privacy statements on what makes them accountable on

the the services they offer. A consumer therefore feels more empowered to assert any faults originating from an automated system than from a human and conditionally has an assurance that they can prevent any violations more systematically.

Eventually a person is more likely to develop a certain kind of trust in a machine more than in a human once the technology is established and widespread. Cars, email, and other gadgets or services we take for granted today are living proof of how inevitably this trust grows, for the better or worse and depending on the degree of affinity to the related technology, aversion or ignorance in a business sector. Trust in a system can grow once it is certified to have little to minimal exceptions. Besides enriching the value chain, it is a key in setting a technology to become an industry standard. Therefore, if a voice assistant is shown to deliver reproducible results disregarding a person's profile, this definitely contributes towards the credibility of the system. This is however not an easy case, since an advanced voice assistant is not expected to be deterministic in most situations, otherwise it becomes boring! We elaborate later in this thesis how we handle this problem.

On the other hand, Information Retrieval Systems range from web pages (e.g. frequently asked questions section (FAQ), forums or a search engine). These are in some cases even less effective than contacting a human as getting the proper information takes a lot of time, or the level of trustworthiness or participation levels in a forum are particularly low, the problem stated is too broad or too specific compared to the answer we are seeking. A user also could come unintentionally across false positives in a search and rely on irrelevant information without knowing. Furthermore, some case-related information might be required to have a proper understanding of a situation or a scenario and provide adequate answers. For Example, if a user would like to know if a certain accessory is compatible with their mobile device, they might need to give a model number, which they may or may not know. Consequently, it is of high interest to maintain a system that could determine all these factors autonomously or with the least possible human interference such that a system supersedes the abilities of the classical Q&A approach.

Internationalization is also another factor to take into account. Since languages differ not only in their vocabulary but also grammar, word and sentence structure (e.g. false friends, nuances, phrases, idioms), developing a voice-first device requires a flexible infrastructure and software stack both able to accommodate these deviations. In that respect, not only is region-specificity important, but also being able to cater for people in a region who do not speak its official language or are residing there temporarily. Especially in businesses where it's difficult to hire skilled foreign-language speaking personnel, a voice assistant can overcome this challenge as it would communicate more

accurately and will not have language problems. The customer is then given the option to avoid an inconvenient experience with the typical scenario with a call centre representatives where neither of both parties understands the other. For those who have minimal understanding of the language, there are possibly also options to provide help in the native language if the user is given the options he/she can answer a prompt with. At the very least, a VUI could still give feedback to the user of wether it understands the language or not, since algorithms for detecting a language are not as complex as answering the question once the language and its dialect (also known as locale) is deciphered. All of which are optimal use cases for ML approaches and algorithms such as term weighting discussed in Section 10.5 based on simpler approaches to localization from an original text, such the corpus-based mapping approach. WordNet is one such example. Finally, giving the user clues on what to answer with is also a helpful tool, as we will also discuss in bot design (Chapter 7)

1.2 The Analyzed Scenario

Berlin.de is an online one-stop-shop for approximately 3,7 million residents of the German capital [3] with thousands of visitors daily for information lookup, appointment bookings and even access to local news. As part of a federal modernization procedure with the help of the German Ministry of Interior, D115 was launched in 2009 [17] as a phone service to help residents find relevant information about a public service or municipality, something that can be tricky if a person has no overview of the local government structure and still not always easy even with the help of search engines nowadays with the array of public services provided. To promote information accessibility, D115 continuously aims at expanding its reach and services. It is therefore worth exploring, how to offer D115 services in a fashion that takes advantage of conversational abilities beyond its human personnel or online city portals.

For now, although local authorities rely heavily on their websites to communicate information to the public, the challenge is mainly finding the right service even with an vague query since most of the time a person requesting a public service is not fully aware of the exact service name in the catalogue of services offered or cannot differentiate between two similar public services or even know if one of them is required in their case. In a metropolis with a high influx of incomers, it is also very likely that certain services are frequently pursued, meaning that helping find the right public service or authority is a repetitive task. In this context, thinking of a voice assistant as a public service could have several advantages, like offloading some traffic from the phone service, getting over the language barrier in the case of non-German speakers, expatriates or simply helping native customers formulate the right wording for a query in a more intuitive way than using a search box.

All of which leads to thinking how providing voice assistants and/or a chatbots for public services could make us reap the benefit of available technologies and extend it aiming for continuous modernisation of our connected living to blend into a seamless digital lifestyle [16].

1.3 Approach and Goals

In the following thesis, we take these factors into consideration and narrow them down to fit our scenario tailored for the e-Government sector. We start by surveying industry-standard voice assistants platforms and frameworks and present a solution that caters for the local municipality of Berlin, Germany. For this we choose to present our solution using the Alexa platform based on our criteria in Section 5.3.

With an agile approach, we present functional code for running and deploying the aforementioned service on Amazon's voice assistant through a so-called ALEXA SKILL (referred to here from here on as 'Skill') and analyse the strengths and weaknesses of choosing this option. We start by exploring information on the public services offered through https://service.berlin.de to understand how to implement our Skill. We then choose a set of exemplary public services use cases with different levels of implementation difficulty to prototype how all current and future Berlin.de's services should be handled using knowledge from an existing database operating the current VIRTUAL CITIZEN ASSISTANT discussed in Section 3. Difficulty is expressed as the required knowledge by the Skill to perform a user's request. This includes for instance the system's abilities to understand the district location intended by the user through only a postal code input.

Further, we test the Skill using some unit tests for the code, runtime scenarios and usability tests (mainly in the form of A-B Tests) for the interaction model especially with the ability to identify and deal with different inputs from various users]. In the implementation, we already have the advantage of being part of Alexa's system to entertain the user with many small-talk abilities which allows us to build on its existing and continually expanding database for question-answer models.

In summary, these goals were defined for the system:

- We want to understand user's requests about public services through different sentence formulations.
- We want to deliver differently relevant parts of information based on the service requested (e.g. cost, processing time).
- We want to communicate with an API endpoint that builds on the service catalogue of Berlin.de's Virtual Citizen Assistant .

- We want to produce a scalable, modular system that can be extended in the future.

Further technical requirements are discussed in Chapter 7.

1.4 Structure of the Thesis

This thesis is organised as follows: In Chapter 2 we talk about use cases of chatbots and voice assistants in the public sector, show related work and introduce comparative examples through e-Government solutions in Vienna, Austria and Georgia, USA. We compare them to the current chatbot application (the VIRTUAL CITIZEN ASSISTANT) available on service.berlin.de [1], which we build our work on with the same underlying technology using Apache Solr as we discuss in Chapter 3.

Going beyond a theoretical analysis of the voice user interface paradigm in Chapter 4 We shed light within the State of The Art context in Chapter 5 on the available platforms in Section 5.2 and explain in the subsequent Section 5.3 why we choose Amazon, going over the idea behind Alexa's Skills and explain the structure of Amazon and Amazon Web Services Ecosystem to operate Alexa in Chapter 6

In Chapter 7 we introduce our design of the Skill and discuss how it addresses redundant boilerplate code, small-talk abilities and briefly highlight some limitations of the system. We also present the frameworks and Application Programming Interfaces (APIs) we use in our own implementation, the prerequisites and artefacts provided initially through a breakdown into several cases simulating top common uses on Berlin's City Portal.

Chapter 9 explains our implementation solution. We elaborate in particular on how we deal with individual public services in the interaction model from front-end and back-end perspective, how we connect the web services together and analytically document the code structure, endpoints in addition to the deployment process.

The related implementation code is available on GitHub [2] with a digital copy attached to this work.

Chapter 10 evaluates our implementation with respect to our target in the defined use cases. We base our evaluation models on a survey we conduct that helps us specify requirement details, as well as automated tests provided by the framewor we use in addition to usability, performance and unit tests.

In Chapter 11, we conclude the work focusing on how our implementation is attainable through Alexa and where future works can be directed for an e-Government solution.

[1] `https://service.berlin.de/virtueller-assistent/`
[2] `https://www.github.com/megantosh/AlexaImAmt`

Chapter 2

Background and Related Work

As our motivation stems from the continuous worldwide trends in digitalisation, we discuss in this chapter current application categories with respect to chatbots and voice assistans and talk about governments' efforts in keeping up with these trends as part of their modernisation stream in the age of social media, Internet of Things (IoT) and the interaction between both. We then show relevant examples in our context of voice assistants.

2.1 Topology of Chatbots and Voice Assistants

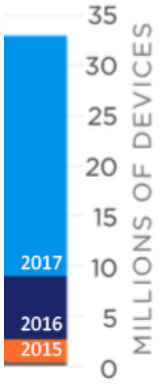
Chatbots and voice assistants come in numerous mouldings; on the web, in apps, through smartphones or as a combination of those and beyond. In Barot and Oren [45] we observe in which different purposes the industry caters for promoting chatbots as a replacement to apps and the voice-first design. As we witness the integration of chatbots in retail, productivity, entertainment and create more possibilities to integrate them in our daily lives, it is obvious that the trend will persist for a while before the market is saturated. Unlike app marketplaces like Apple's App Store and the Google Play Store which have become a commonplace term despite their slow stagnation in the recent years, chatbots and voice assistants are predicted to remain trendy until at least 2020 as per VoiceLabs predictions.

Moreover, SearchMetrics' upcoming report is expected to include a a dedicated part to voice searches as the analytics company advises on SEO tweaks for Google Voice Search [21] and argue how voice assistants are becoming a ubiquitous market. In fact, with hundreds of thousands of chatbots only on Facebook and tens of thousands of voice assistants only on Alexa the two major platforms are serving for an array of categories. Through these, leisure becomes possible by playing a game [44] or making a holiday booking through a chatbot just as productivity does not get any less serious with another bot taking appointments. Although their markets differ, chatbots and voice assistants are

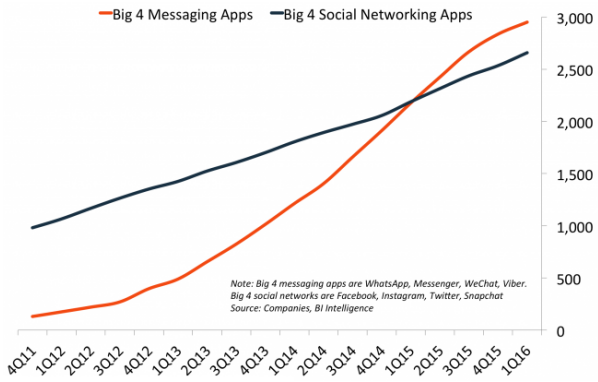
becoming prominent around the same concept of doing or delegating one task without the need to leave the same interface or requiring additional resources (apps or even a screen).

Figure 2.1: Trends of increasing use of voice as an interface are shown with an increasing in number of devices and user base in the last seven years. These and further statistics [1] show the necessity of developing for voice.

(a) Voice-First Device Footprint [2]



(b) Global Monthly Active Users in Millions [25]



2.2 Digitalisation in the Public Sector

There is no doubt that social media has changed the shape of our society in the last twelve years. Not only by affecting the way we communicate, but also by generating immense amounts of data, creating new jobs and disrupting economies en masse. Meanwhile, governments are continuously striving to market their image in more creative ways than ever to target more segments in society. From politicians' tweets being a novelty to now making headlines, The sudden interest of governments in social media platforms can be seen as an attempt to become more transparent and reach more public awareness. It is more likely for instance that people from narrower fields of interest or of wider age range would find out about a public figure by follwing them and finding them on their feed rather than by reading news about them on dedicated news website. This now goes as far as country leaders curating stories on Instagram and existing on platforms that are not only for the politically active.

This is ultimately because the social media platforms, as part of many new technologies are designed with the focus of reach in mind much more than only availing information to the target. We distinguish between the pull-based model of information

[1] e.g. `http://www.kpcb.com/blog/2016-internet-trends-report`

retrieval, where a person would actively look for a certain piece of information usually through looking it up through search, vs. the push model, where the information is 'thrown' to the user without them actively asking for it, usually by subscribing to the source or news outlet and more likely through audiovisuals than trough text. Unsurprisingly, such platforms become a key player in redefining fame, identity, image and even cognitive behaviour.

The push-model is a new strategy to dissemination of knowledge, which local and national governments embrace [27] to bridge the gap in their outreach to the public. Although the digitalisation trend in governments generally emerges from the desire to expand infrastructure for information exchange and availing publicly accessible data (e.g. open government data) [46] or classified data (e.g. the Schengen Information System [2]), digital media has now become an integral part of it, too, making it inseparable from many institutions' compositions and their decision-taking process.

The amount of digital media we use daily everywhere would not be possible without having an expansive digital infrastructure. In Germany, many advanced milestones in digitalisation were attained in the last years. From eliminating paper in many administrative workflows to funding research or successful businesses including start-ups in high-tech fields and seeking renewable energy as a foundation to power the exponentially growing number of devices in the country with a reduced carbon footprint, the aspect of digital media has been relatively neglected compared to other countries.

Figure 2.2: Top Ten Countries for E-Government Development Index in Europe based on [46]. Germany is 8th in the list with a decreasing development rate compared to previous years (lower range within EU countries). High rank drop tendency as opposed to increases at counterparts (NL, FR)

Country	Region	Sub-Region	OSI	HCI	TII	EGDI	EGDI Level	2016 Rank	
United Kingdom	Europe	Northern Europe	1.0000	0.9402	0.8177	0.9193	Very High	1	
Finland	Europe	Northern Europe	0.9420	0.9440	0.7590	0.8817	Very High	5	
Sweden	Europe	Northern Europe	0.8768	0.9210	0.8134	0.8704	Very High	6	
Netherlands	Europe	Western Europe	0.9275	0.9183	0.7517	0.8659	Very High	7	
Denmark	Europe	Northern Europe	0.7754	0.9530	0.8247	0.8510	Very High	9	
France	Europe	Western Europe	0.9420	0.8445	0.7502	0.8456	Very High	10	
Estonia	Europe	Northern Europe	0.8913	0.8761	0.7329	0.8334	Very High	13	
Germany	Europe	Western Europe	0.8406	0.8882	0.7342	0.8210	Very High	15	
Austria	Europe	Western Europe	0.9130	0.8396	0.7098	0.8208	Very High	16	
Spain	Europe	Southern Europe	0.9130	0.8782	0.6493	0.8135	Very High	17	

*Note: The Ranking Trend lines display the country rankings, with 1 being the top ranked and appearing at the bottom of the vertical axis, and 193 being the lowest ranked and appearing at the top of the vertical axis. Therefore, the lower is the graphical point, the higher is the ranking. The horizontal axis represents the survey periods of the *UN E-Government Survey*, i.e. 2003, 2004, 2005, 2008, 2010, 2012, 2014 and 2016.

[2] https://en.wikipedia.org/wiki/Schengen_Information_System

According to numbers from the United Nations' E-Government Development Index in 2016, Germany ranked 15th worldwide and 8th within Europe with an average of 0.82 out of a maximum of 1 , substantially outranked by the United Kingdom, the United States, Singapore and France. Outperforming its previous statistics from 2014 where its index scored 0.79 with a world average of 0.47, it was also outranked by Austria, Israel and Bahrain [5] in addition to the aforementioned countries as listed in Figure 2.2. Germany's ranking went up from 21st to 16th worldwide in the last two years.

the E-Government Development index (EGDI) is composed of a calculation based on proportionally normalised averages of the Online Service Index (OSI), the Human Capital Index and the Telecomunication Infrastructure Index. When it comes to engagement through e-participation, Germany performance lies also in the top quartile. Its Online Service Index performed 21st worldwide (Figure 2.3), which shows the continuous strive to maintain relatively many rights to access government information online and an open data government policy on the web.

Figure 2.3: Online Service Index for Top 30 Countries Worldwide based on The UN Department of Economic and Social Affairs [46] with Germany ranking 21st in 2016

In an age where we take the internet for granted, we also expect to have multiple communication channels open to government representatives and public authorities. Hence, out of public interest, governments are advised not only to focus on digitising their internal structure, but also their representation on the public sphere. Especially at times of change like with the introduction of autonomous driving and the regulations related to it, transparency is a continuous expectation from the people. As stated above, since transparency heavily relies on public presence, which in turn depends on availing more communication platforms, voice assistant services are definitely worthy of consideration in that context.

2.3 Evaluation of Governments Presence in the Cloud

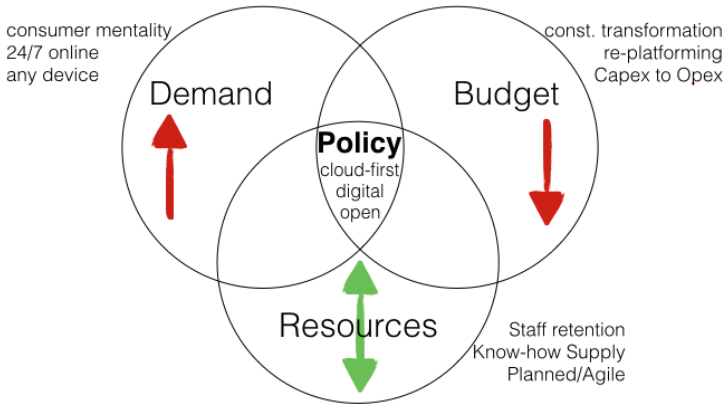
With fast-paced technological advancements, advocacy for a fair use policy is equally important to the consumers as to the industry responsible for spreading them. Raising

awareness about the necessity for legal adjustments plays a determining role in policy-making and governance of ICT businesses. As we experience how personal data has become a valuable commodity to enterprises, advocating for tightening regulations to avoid misuse of such sensitive data has become crucial over the last years. In the search for methods of circumvention to new kinds of monopolies, cybercrime as well as other frauds and immoral breaches, several governments worldwide have addressed the potentials and threats of data collection for Information Systems in use with respect to different local contexts.

While passing laws and regulations is a positive action from a general perspective, it is not always easy to guarantee that these will not impede the advancements in favour of keeping a stable status quo or encourage other motives. In many MENA countries, Turkey and Iran for instance, we witness internet censorship in the form of website blocks with no legal basis. Another alarming example is the case of Egypt's laws on cybercrime for it prohibits the use of cryptocurrencies to the public while encouraging its own government to make additional revenues through Bitcoin and ad traffic networks simultaneous to spying on the people e.g. by intercepting the HTTPS protocol nationwide [31]. Reports from Privacy International [20] and The Association of Freedom of Expression [32] shed light through different incidents related to the Egyptian Cybercrime Law on how it can have a negative impact to regulating technology.

As there are very few international regulations, it remains up to each federation or alliance of countries to handle the case of data privacy individually, which can be a tough job. And though positive action should be credited for the attempt, there are still many decisions to take national and International level to secure an ethical handling of personal data.

Figure 2.4: In a common government Scenario, increasing demand and degreasing budget push against each other and require flexible resource allocation solution. With all these factors considered, a cloud-first approach can combine all elements in the policy-making decision. Based on Bockelman in [43]



When it comes to governments' operations in the cloud, we see different degrees of aversion to anticipating this relatively new concept. As cloud infrastructures can be quintessential in scalable operations, governments can be among number one users of cloud microservices, especially with demand variability in peak times that are not always easy to predict (Figure 2.4). Among the many benefits of cloud services, scalability is arguably the most convincing. By that we mean the ability of an operator to allocate resources or terminate them as needed to avoid extra costs and infrastructure maintenance. Consequently, operating as a government in the cloud and outsourcing the digital infrastructure is an incentive to massive cost reduction. Additionally, as governments operate on strict budgets, cloud operations can accommodate much easier to variable expenses instead of waiting for long-term investments in computing infrastructure [43]. Such make-or-buy-decisions have great impact to the way services are offered, as they allow higher degrees of specialisation, which in turn make further services only possible through externalising resources. Further, using off-premise resources enables governments to pilot more project and adjust to the public demand accordingly. This means that services can be expanded or taken down based on how they good or bad they are perceived by their end-users, be they citizen or the government itself. The UK Government G-Cloud Framework[3] is one such examples that shows a harmonious cloud operation of government services.

If we consider our presented solution of a voice assistant as such service, hosting in the cloud is considered as the most desirable solution to test performance and adaptability such that we are able to up-or downscale resources for users in real-time. Yet, the trade-off of cloud computing remains the data protection issue as infringing such policies cannot be tolerated by the public and could have drastic consequences on legislation.

From EU perspective most recently, the European Commission enforces several adjustments in spirit of adhering to the General Data Protection Regulation (GDPR) effective May 2018. Of these adjustments, the new norms facilitate a better delegation of responsibility to the operation of the cloud host as well as to the services hosted on it. This comes along with digital infrastructure certifications that can be obtained to give more securities to customers on a microservice, from Software as a Service (SaaS) to Platforms or Infrastructures as a Service (PaaS, IaaS). Among such certifications in Germany is the Cloud Computing Compliance Controls Catalog (C5) Standard, to which AWS as a top cloud microservices operator for example complies to [4]. Since compliance is an major factor in choosing the cloud operator, the company lobbies for its compliance through numerous formats [5].

In short, It is difficult to imagine deploying a competitive service today on a large

[3] https://www.gov.uk/guidance/the-g-cloud-framework-on-the-digital-marketplace

[4] https://aws.amazon.com/compliance/bsi-c5/

[5] https://aws.amazon.com/compliance/germany-data-protection/

scale without utilising the power of cloud computing. Governments have extensive resources to inform themselves on how to approach their utility best from cloud operators and can enforce norms that cater to good ethics of data use.

2.4 Worldwide Examples of Bots in E-Government

Knowing that the Online Service Index of a country consists of the services available to different groups of society, we can argue that vulnerable groups (i.e. the poor, persons with disabilities, older people, immigrants, women and youth) can be easily marginalised despite a high index. Another statistic by the German Federal Statistics Office (Statistisches Bundesamt) shows that most online public services in 2016 were targeting persons of age 25-44 with an equal offer of services for age groups 45-67 and 16-24 with the least amount of services targeted at youngsters between age 10 and 15 comprising 15% of the Federal Catalogue of Public Services (Leistungskatalog, LeiKa) [41]. For that reason, we acknowledge the importance of inclusion especially towards vulnerable groups in the process of expanding a online public service.

This gives us the advantage to promote for inclusion in the development process of a chatbot or a voice service assistant especially since the domain is relatively still uncharted. As Zehlike et al. conclude in their research about algorithmic bias and discrimination discovery, given the possible disparity measures ML algorithms can result in [28], a holistic approach is required to have a balanced distribution of services in different fields (e.g. Health, Finance, Administration) for different groups of society (protected and unprotected groups). Already offering a service on platform that offers less constrains to the user than a browser's graphical user interface (GUI) allows per se more options for inclusion but could also be limiting of the system reacts to a vocabulary used only by a certain group for instance.

In terms of functionality and testing of a chatbot or voice assistant system on inclusion metrics, WitLingo [6], a company specialised in voice-driven apps for public and private sector enterprises believes in that sense that "getting feedback [on performance of a voice service to different groups] is more important than measuring ROI" [35] with a report by Voysis / Retail TouchPoints [47] reflecting this statement in the retail sector.

While there are several cities such as Singapore and Los Angeles integrating chatbots and voice assistants into their administration's websites and coming up with all sorts of interactive solutions, for this research we choose the City of Vienna chatbot service WIENBOT[7] given that it uses German and the Alexa Skill GEORGIAGOV[8] for a comparison in US English, as we examine both languages in our development process.

[6] `https://www.witlingo.com/`

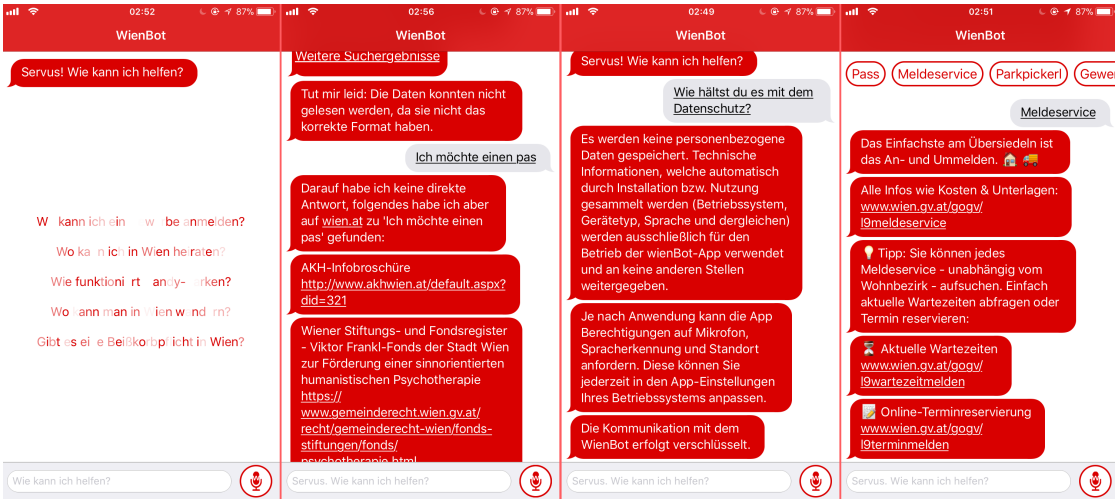
[7] `https://www.wien.gv.at/bot/`

[8] `https://www.amazon.com/GeorgiaGov-Interactive-GTA-Ask/dp/B074XBQGTQ`
Demo on `https://vimeo.com/216737044`

WienBot

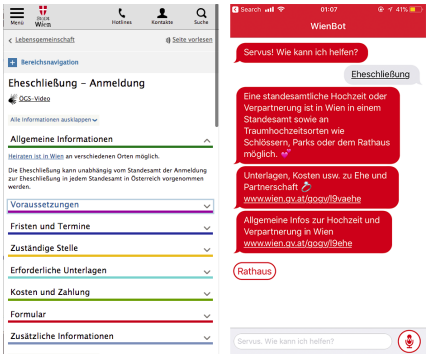
Started out of an amateur project as a Facebook Messenger bot and officially launched in December 2017, the chatbot is developed by and for the municipality of Vienna, Austria with accessibility in mind, for which reason the decision was to create a standalone application. and was the first official chatbot made for a city council. It is famous for its Viennese accent and its witty, verbose (and emojiful, Figure 2.5), yet polite responses in the German'Sie' form.

Figure 2.5: Screenshots of Conversation with WienBot showing the verbose interaction.



With over 200 so-called 'stories' in February 2017, the chatbot started as a query database for the top 100 FAQ. A first look on the Vienna government website beyond the chatbot gives a very complex and crowded impression. This is due to the range of services offered through the centralised website in addition to the text-rich content on all pages. The first clear structure prevails as we get into a specific service. On the German version of the website (the English version has only hyperlinks), following tabs are consistent:

- Public Service Name
- General Infomration - *Allgemeine Informationen*
- Prerequisites - *Voraussetzungen*
- Deadlines and Dates - *Fristen und Termine*
- Responsible Authority *Zuständige Stelle*
- Required Documents - *Erforderliche Unterlagen*
- Costs and Payment - *Kosten und Zahlung*
- Form - *Formular* (includes online reservation links)
- Additional Information (Zusätzliche Information)



Asking for the same service on the chatbot, we get a different overview as shown.

GEORGIAGOV

Developed for the State of Georgia, USA by Acquia[9] the voice assistant is an Alexa Skill. We introduce Skills later in Section 5.2 and discuss them in detail starting Chapter 5. For now, it is enough to know that an Alexa Skill is comparable to an app on smartphone that simply extends its functionality and that to operate the Skill an Alexa-enabled device is required.

> The state of Georgia has always been on the forefront of web accessibility. For example, from 2002 until 2006, Georgia piloted a time-limited text-to-speech telephony service which would allow website information and popular services like driver's license renewal to be offered to citizens [13]

While we are less interested in the content delivered through the voice assistant and have no access to the GeorgiaGov Skill (at some point known as AskGeorgia) from the German Amazon Store, we use the US store for our conversation testing purpose to reverse-engineer the relevant design of dialogue flow and eventually pick what is helpful to our own implementation.

Here are a few outtakes of a conversation with the Alexa Skill GEORGIAGOV:

" Alexa, open Georgia Gov "

" Welcome to Georgia Gov. Feel free to ask a quesion "

" How can I renew my license? "

" Marriage licenses. Does this sound helpful? "

" No "

" How about this: Nursing licenses "

[9] https://www.acquia.com

“ Not really! ”

“ You can call 1-800-GEO-RGIA [...] over the phone. If you have another question, feel free to ask it. Otherwise, you can say exit. ”

“ How do I get food stamps? ”

“ Food stamps. Does this sound helpful? ”

“ yes ”

“ The food stamp program provides monthly funds for families [...]. As of March 2016 [...]. I have a list of related information. Would you like to hear it? ”

“ yes ”

“ “How do I report food stamp fraud”. Does this sound useful? ”

“ No ”

“ That’s the last on the information on this topic. If you need a phone number for further assistance, say “Give me their phone number”. Otherwise, you can ask another question or say “exit”. ”

“ Exit! ”

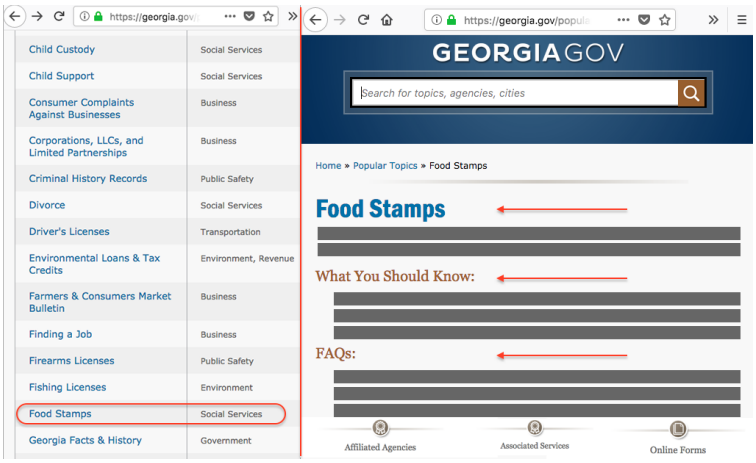
We compare the Skill to the information on `www.georgia.gov` and realise that Alexa mainly provides a narrowed down version of the website. In some cases, a few spoken paragraphs are very similar in content but not exact to the text on the web page. Noteworthy is that Georgia.gov provides PDF and online forms that can be submitted through the website or domains affiliated with it. In terms of capabilities, the Skill

provides objective facts only and no content is customised to the user's profile (age, sex or location).

As mentioned on the Amazon website, the content data is dynamically generated. After familiarising ourselves with Alexa, we understand this means that the Skill is connected to an API not hosted through Amazon, giving the maker of the Skill the ability to change the information delivered through the Skill without Amazon's permission (more on permissions in Sections 5.2 and 9.6). The conversation flows dynamically, with Alexa engaging the user by asking them questions or listing options to which they need to answer with a 'yes' or 'no' to hear information on public service topics that to our knowledge listed on the popular topics section of the website [10]. A view on the sitemap [11] shows that there are no other topics than the popular ones at least on the root level.

Each of the available topics is associated with one or many departments and relates to a few search terms (Figure 2.6). By navigating on the website to any of the topics, we see a clear division between a "What You Should Know" section and another "FAQ" section. At the bottom of the page, affiliated agencies, online forms and associated services are listed. The Skill reflects the same structure through conversation. The user starts a dialogue within the Skill with a topic exact or synonymous to any of the listed ones. Once Alexa determines the topic, general information similar to the heading is given, then Alexa prompts the user if they want to know about each item in the list of related services, then ends the dialogue about this topic by asking the user if they want a phone number of the associated department. The user can then ask another question related to the same or another topic. Of course some options are defined within the conversation flow. In Chapter 10 we show in Figure 10.1 a more detailed flow of the conversation.

Figure 2.6: Georgia.gov Popular Topics - Based on georgia.gov/popular-topics



[10] https://georgia.gov/popular-topics
[11] https://georgia.gov/sitemap

Chapter 3

Berlin as a Use Case

As Schwarzer et al. state, few systems are available that "enable Berlin's citizens to get informed about governmental services" [26]. As part of the modernisation process, the VIRTUAL CITIZEN ASSISTANT - *Virtueller Bürgerassistent* was born in the Information Retrieval and Machine Learning Competence Centre of the Technical University of Berlin's Distributed Artificial Intelligence Laboratory.

This chapter explores the running software of the Virtual Citizen Assistant in detail to analyse which artefacts can be reused. It also explains the context of the software from legal and operational perspective.

3.1 D115

The German Ministry of Interior which operates the national hotline for citizen services D115[1] argues in its whitepaper on the service [17] that even with the widespread of internet use in the country, a telephone hotline is a valuable service. 70 % of the D115 users use the internet according to a survey by the entity (2013). One of the main advantages of the service is that it offloads the lookup of information from the customers to its employees and guarantees that the customer will get the right answer concerning questions related to public services. Depending on the service, appointment bookings can be arranged (sometimes exclusively) through the hotline. Effectively, this means relying more on manual labour, which has a twofold impact for an administrative entity in the government. On one hand, D115 opens jobs in the labour market for public service employees as call centre representatives, strategists, editorial boards throughout Germany inter alia. For a government entity this speaks positively in terms of reducing unemployment rates. On the other hand, with a budget coming from taxpayers, there is a need to choose how to allocate this budget dynamically in a fashion that does not come to the disadvantage of societal segments.

[1] `http://www.115.de`

Despite the heated worldwide debatable about technological advancement eradicating job opportunities, if we advocate to offer services compatible to serving users needs, the necessity to diversify the service catalogue remains prevalent. Including online services as a possible alternative, at least to include minority groups with hearing or visual disabilities is inarguably an important step towards a higher Online Service Index and EGDI respecitvely.

The whitepaper states a few expectations of the D115 service. Taking its promises into account, we set the following goals to our service:

1. 24/7 reliable availability in accordance with underlying Service Level Agreements (SLAs)

2. Avoiding the need to visit a municipality office if possible

3. Clear explanations on what a public service is about, how it is performed and all necessary detail around the public service to make it conform to the Right to Information (Article 19 of the Universal Declaration of Human Rights) in accordance with the German Freedom of Information Act - Gesetz zur Regelung des Zugangs zu Informationen des Bundes [2].

4. Continuous improvements to the voice service offering through learning from new specific cases that can be helpful to solve others in the future.

5. An implementation that tries to help and not repel customers.

6. A service that is friendly and respectful of its target audience

7. An open option to equip customers with more information by triggering them to ask for, in case it could be helpful without them knowing at the time of inquiring.

8. A service that gives facts anonymously to the customer without collecting data that can potentially result in any information bias at the time of service or in the future

9. A flexible service that can respond to the questions answered through analysing the relevant information and omitting unnecessary details.

With the Virtual Citizen Assistant as our starting point, we look at how these expectations are fulfilled through the chatbot. we then touch in Chapter 7 on how we attain the same goals.

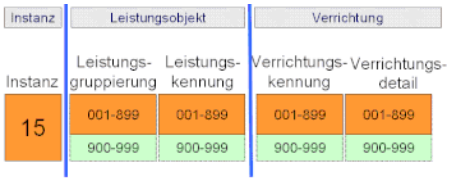
[2] `https://www.gesetze-im-internet.de/englisch_ifg/index.html`

3.2 LeiKa

In the German public administration system, most public services are carried out through each of the 16 States with very few services offered on a national level. As each State has a different bureaucratic structure, considered as the E-Government base for Germany, LeiKa [3] is a service catalogue and a way to standardise how different public organs of each State fit into the same legal framework for federal regulations. For instance, it is a requirement for each resident in Germany to register their current (permanent) address at a local authority. Registration differs based on the resident's legal status, e.g. due to a move of house within Germany, within the same State or a move from abroad. For a person who moves from abroad, further regulations apply based on their right to residence and the duration of the stay in Germany (short-term, long-term, permanent). These laws apply to nationals and foreigners (where another differentiation occurs between EU/EEA citizens and nationals of other countries). In that sense, there are different laws that may or may not apply for the same action (a registration of an address), e.g. Section 17 of The Federal Act of Registration [4] - *Bundesmeldegesetz, BMG* in conjunction with Section 22 of The Asylum Act[5] - *Asylgesetz, AsylG*. How this is carried out is left to each state to determine. In many states a foreigner can register themselves at the aliens office. In Berlin, foreigners are required to register only at a Bürgeramt. In order to make sure that the legal framework is covered through the services in every state, ensuring that the law is applied through an appropriate channel, LeiKa serves as a catalogue to keep track of how and through which organs the law is enforced.

Similar to an IBAN without a checksum, LeiKa's key structure is divided into the following identifiers put together (Key System 2.0, Figure 3.1):

Figure 3.1: Leika Key System 2.0, based on [6], ©CC, Wikipedia



- Instance - *Instanz*
- Service Object - *Leistungsobjekt*
- Service Grouping - *Leistungsgruppierung*
- Service Identifier (ID) - *Leistungskennung*
- Performing Entity - *Verrichtung*
- Performing Entity ID - *Verrichtungskennung*
- Performing Entity Detail - *Verrichtungsdetail*

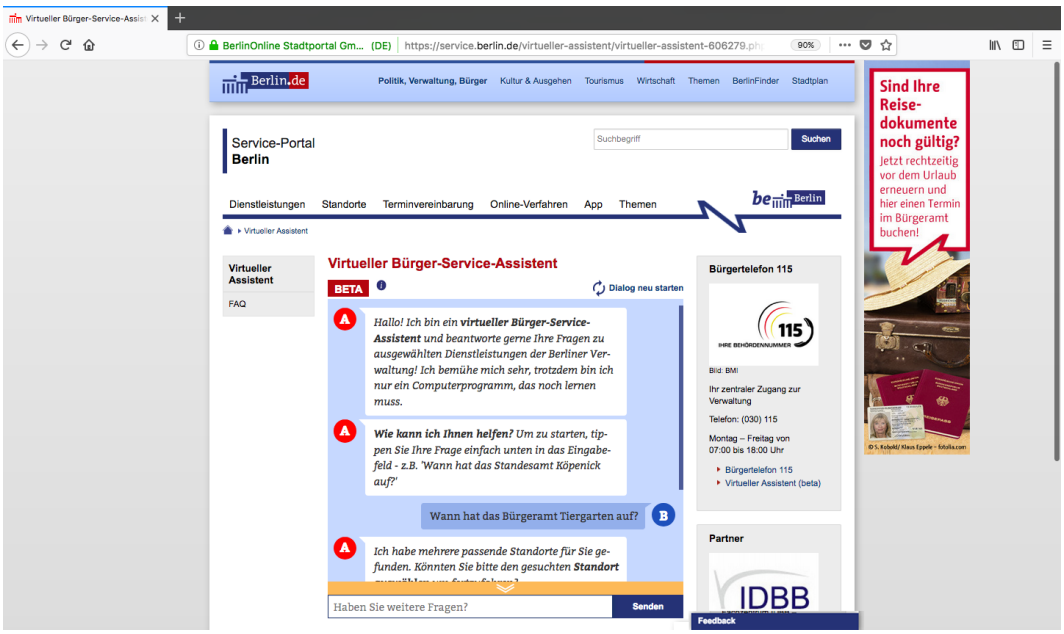
Leika Keys are part of the data provided to the Virtual Citizen Assistant's back end and help us

[3] https://de.wikipedia.org/wiki/LeiKa
[4] https://www.gesetze-im-internet.de/englisch_bmg/index.html
[5] https://www.gesetze-im-internet.de/englisch_asylvfg/index.html
[6] http://www.gk-leika.de/

carry out an implementation not only for Berlin through its universal identification system using serial numbering.

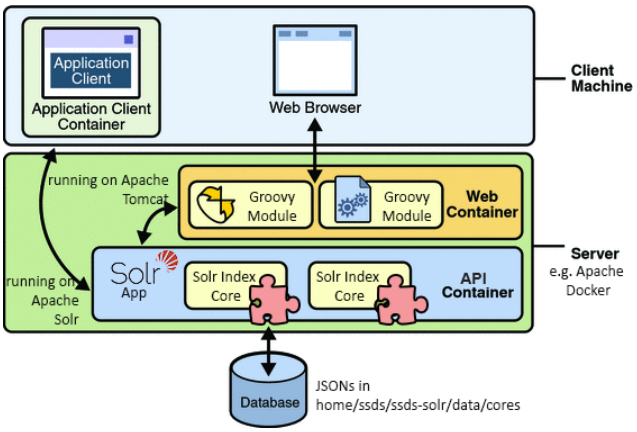
3.3 Understanding The Virtual Citizen Assistant

Figure 3.2: Screenshots of Startup Interface of the Virtual Citizen Assistant showing the initial interaction in a web browser.



The Virtual Citizen Assistant on Berlin.de starts with a greeting and an input field as seen in Figure 3.2. Based on the question text entered by the user, a query goes to the Solr back-end and the responses from there are processed to deliver available options, which the user can choose from. Once a question is answered, the chatbot continues to ask if the user is interested in knowing more about sub-topics related to the service, such as cost or processing time. The chatbot is a rich implementation with many features, e.g. the ability to decipher locations based from input postal codes (Postleizahl, PLZ) and auto-correct among many others. We use such features as an orientation on how our voice assistant should be implemented and what inputs and outputs it needs to support. As we anticipate a voice-first approach, we also demonstrate how these should or should not be adapted in Section 8.1. As such, the logic behind an auto-correct feature for instance has to be re-engineered since spoken word is different than written text.

Figure 3.3: Virtual Service Assistant - based on [7]



By deploying an instance of the chatbot locally, we understand that it is built using Java for the Solr server implementation and Groovy, a Java-syntax-compatible OOP language, to operate a web app on top of Solr as an API to handle user inputs and display answers formatted inside a web browser (e.g. links for navigation in the conversation and standard text). We also understand that the implementation is designed to include States other than Berlin. At the time of this research, another chatbot was in the making for the City-State [8] of Hamburg (parallel to continuing development on Berlin Virtual Citizen Assistant). Both chatbots were able to connect with another partition of the web app that can serve in other domains if the user has questions beyond the scope of public services. A nutrition bot is as an example. Domains other than Berlin are not relevant to our work and are excluded from our scope of analysis.

3.4 Building a Voice Service on The Virtual Citizen Assistant

In the Berlin domain, we narrow down our compiled choice of files for reuse. At this initial stage we cannot depend on reusing code unless we can tailor our platform, yet to be chosen at this stage, to do so. This turned out to be partially applicable as our choice of platform in Section 5.3 shows that we can mostly make the best use of data-containing files relevant to our use-case rather than scripts of application or business logic. Such data files exist in JSON format for the Virtual Citizen Assistant. Data is meant here as content our voice assistant can operate with, ranging from names, URIs to addresses. Complex data types beyond integers and strings needed to be redefined for the new system to understand them. Java and Groovy code were not considered at this stage since we anticipate a different programming language as per Section 5.3. Since JSON is an industry standard, we look for possibilities to parse this information into a system that can read them in and be structured to understand the file format.

[8] https://en.wikipedia.org/wiki/City-state

[8] some graphic elements adopted from Oracle Java Tutorials - https://docs.oracle.com/javaee/5/tutorial/doc/bnabo.html

Starting with `./ssds-data-berlin/json/dienstleistungen/dienstleistung.json` (the file has been reconstructed since this version to be included as a part of the Solr index, i.e. a core), we analyse the content of a few fields (as JSON objects / nodes) in Figures 3.4 and 3.5 to draft Tables 3.1, 3.2 and 3.4. We introduce the remaining Solr index fields below.

Figure 3.4: Primary Nodes of `Dienstleistungen.json` including metadata about each service, e.g. processing time and cost.

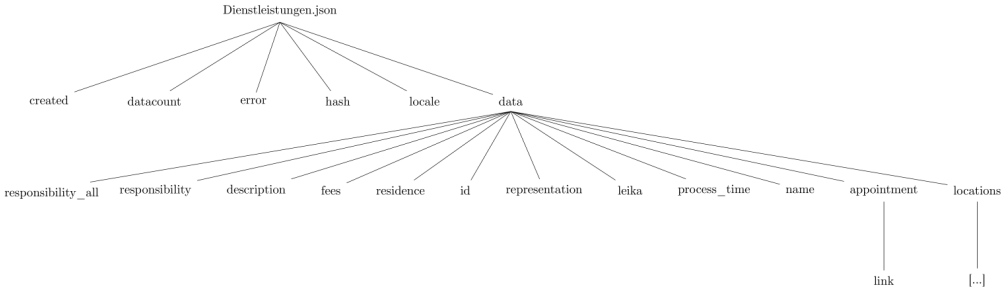
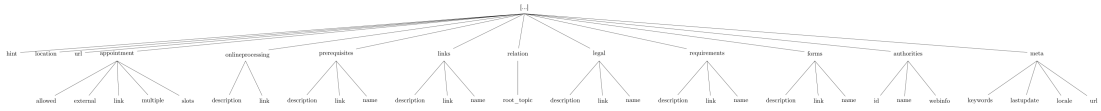


Figure 3.5: Secondary Nodes of `Dienstleistungen.json` show relevant follow-up nodes to the primary node `locations`



From back end side, the files mentioned above reporesent cores in a Solr index. The latest version of the underlying Solr server (May 2018) comprises the following cores:

d115

An index core designed to query the latest public services available on Berlin.de. Example services: "*Personalausweis beantragen* - issuing a passport", "*Blindenhilfe* - monetary relief for the blind".
This core initally included 616 nodes at the beginning of this research and now varies between 670 and 680 services depending on the original LeiKa. We will use this core later on as our primary API.

d115Extern

An index core designed to include external Hyperlinks outside the service.berlin.de subdomain.

Examples: "*Gentechnik* - gene technology" routes to `https://www.berlin.de/lageso/gesundheit/gesundheitsschutz/gentechnik/`

d115Locations

An index core designed to query locations of authorities' offices offering a public service and their opening hours
e.g. "Kfz-Zulassungsbehörde-Lichtenberg"" the authorities' offices and their opening hours.
Based on (`./ssds-data-berlin/json/behoerden/behoerden.json` which will help us indirectly in our implementation later on.

d115Spelling

An index core designed to make the chatbot able to determine spelling mistakes in names of entities (e.g. offices, services) and automatically correct these in the conversation flow

d115Topics

An index core designed to query the database for topics. Each topic collects a set of services that are of the same interest group. For instance, a topic "Ausländerangelegenheiten" comprises all sorts of visa and residence permit types. Another topic on "Auto und Verkehr" includes services like parking options, paying fines and issuing a driver's licence.

smalltalk

An index core designed to make the chatbot more interactive by including simple questions and answers to them not related to Berlin.de's service catalogue

Tables 3.1 and 3.2 represent the Solr index cores available on `/home/ssds/ssds-solr/data/cores/` and show the fields (child nodes) of each core on Solr, as it is important to have the structure of these files for our implementation.We refer to our Solr instance from now on as the API, since it is built and operates independently from the Groovy web app and only delivers the results to it. Hence, it offers an opportunity to be shared by another programme, rendering it into a standalone API endpoint that can be hosted on its own server as will be discussed in Chapter 9.

In Table 3.4 we cover an excerpt of most node elements we will later use in a similar structure for our own front- and back-end.

After understanding the links between the API and the sitemap of the Berlin City Portal, we draw a few connections summarised in Table 3.3 to understand how we can proceed with the API and eventually modify or expand it with additional parsers such as Apache Nutch or other runtime parsers (e.g. to find the next appointment, since this is not provided by the API) to suit our voice service solution.

Table 3.1: Structure of API - Solr cores 1-5, version (*ver.*) and relevance (rel.) to our implementation. **Boldfaced entries** are explained in Table 3.4

Core Name, *Ver.*, rel.	No. Entries	Fields for each entry (by Order of Child Nodes)
d115 *426,* very relevant	672	**id** **d115Url** **d115Name** ssdsAll **d115Description** d115Synonym d115Position **d115InfoLaw** **d115Prerequisites** **d115Requirements** d115Forms **d115Fees** **d115ProcessTime** d115AppointmentLink d115ServiceLocations d115ServiceLocationsJson d115ServiceResponsibility d115ServiceResponsibilityAll d115OnlineProcessingLink **leikaId** leikaName leikaGruppe leikaKennung leikaVerrichtung leikaVerrichtungDetail leikaSynonym ssdsName ssdsLemma ssdsLongName ssdsGruppe ssdsGruppeDict ssdsKennung ssdsKennungDict ssdsSynonym ssdsSynonymDict _version_
d115Extern *396,* semi-relevant	1488	id d115Topic d115Name d115Link d115Keywords _version_
d115Spelling *678,* semi-relevant	1233	id complex object (irrelevant) _root_ _version_
d115Topics *426,* semi-relevant	167	id d115Name d115Path d115Keywords d115Links _version_
smalltalk *396,* semi-relevant	1366	id smalltalkAnswers smalltalkQuestion _root_ _version_

Table 3.2: Structure of API (cont'd) - Solr cores no. 6 (last), version (*ver.*) and relevance (rel.) to our implementation.

Core Name, *Ver.*, rel.	No. Entries	Fields for each entry (by Order of Child Nodes)	
d115Locations *426,* relevant	560	id	d115Url
		d115Name	d115LongName
		d115Type	d115SuperType
		d115District	d115SuperDistrict
		d115AddressHouseNumber	d115AddressCity
		d115AddressStreet	d115AddressPostalCode
		d115AddressGeo	d115AddressNote
		d115AddressUrgentEndDate	d115TransitTram
		d115AccessibilityElevator	d115AccessibilityAccess
		d115AccessibilityParking	d115AccessibilityWc
		d115ContactEmail	d115ContactFax
		d115ContactPhone	d115ContactWebInfo
		d115ContactCompetence	d115ContactSignedMail
		d115ContactSignedMailLink	d115Payment
		d115PaymentCode	d115AppointmentNote
		d115OpeningTimesMonday	d115OpeningTimesTuesday
		d115OpeningTimesWednesday	d115OpeningTimesThursday
		d115OpeningTimesFriday	d115OpeningTimesSpecial
		d115LocationServices	d115LocationServicesJson
		version	

Table 3.3: Structure of Berlin City Portal URLs. We use these to collect additional information beyond the API's content (i.e. Solr fields).
Base URL: `https://service.berlin.de/`

Web Page displays	URL Path	Suffix
a public service	dienstleistung/	`id`
an authority homepage	behoerden/	variable, no base URL, external link
a service location	standort/	`d115ServiceLocation`

Table 3.4: Relevant nodes in Solr API query results in core `d115`, based on `Dienstleistungen.json` usable in our implementation - at the example of a public service called: "Fiktionsbescheinigung"

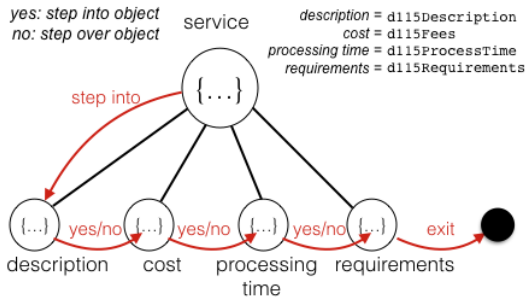
Key<Type >	*[Omitted Text]* in *Example Value* — Description — **Remarks on Syntax**
`id` `<int>`	*326233* public service ID
`d115URL` `<string>`	*https://service.berlin.de/dienstleistung/326233/* link to public service URL **URL structure includes `id` - See Table 3.3**
`d115Name` `<string>`	*Fiktionsbescheinigung* public service name as listed on Berlin.de
`d115Description` `<string>`	*"Eine Fiktionsbescheinigung wird ausgestellt, wenn über einen [...]"* Introductory paragraph about the service web page **incl. HTML tags**
`d115InfoLaw` `<string>`	*§81 Aufenthaltsgesetz - AufenthG ::: false ::: http://www.gesetze-im-inter[...]* Legal base for offering this public service
`d115Prerequisites`	*"Persönliche Vorsprache ist erforderlich ::: false ::: ", [...]* Conditions a citizen needs to fulfill in order to obtain the service **incl. non-fluid text (unique patterns, parsed/encoded HTML)**
`d115Requirements`	*"Bisheriger Aufenthaltstitel ::: Soweit vorhanden, ist der [...] ", [...]* Required document(-s) citizens need to present to obtain service **incl. non-fluid text (unique patterns, parsed/encoded HTML)**
`d115Fees` `<string>`	*"Ab dem 01.09.2017: [...] "* Cost of the service **incl. non-fluid text (unique patterns, parsed/encoded HTML) variable free Text for same value, e.g. free services cannot be explicitly read as primitive data type** `int/bool`
`d115ProcessTime` `<string>`	*"Die Fiktionsbescheinigung wird bei Vorsprache ausgestellt."* Time required to process the application for the service **incl. non-fluid text (unique patterns, parsed/encoded HTML) variable free Text for same value, e.g. immediate services cannot be explicitly read as primitive data type** `int/bool`
`leikaId` `<int>`	*99010008012000* LeiKa identifier as discussed in Section 3.2

There are other elements not directly available in the Solr index as fields such as objects from `./ssds-data-berlin/json/districts.json`, which displays

each municipality - *Bezirksamt* in relation to the districts - *Ortsteile* and Postal codes - *PLZ* it contains. As these are helpful to our implementation structure, they make good candidates for a new object type (as discussed later in the interaction model, Chapter 7). There are some naming ambiguities in the file structure that will need an implicit type inference. For instance, the type of location is only distinct through the name, e.g. 'Lichtenberg' for the municipality and 'Lichtenberg-sub' for the district.

3.5 The Virtual Citzen Assistant at Runtime

Figure 3.6: simplified demonstration of how nodes are traversed in Virtual Citizen Assistant



The web app revolves around the following algorithm: once a public service has been caught from the user's input, the chatbot shows the description of the service, followed by a series of questions beginning with a yes/no one to determine if the user is interested in knowing more about the cost of the service, followed by another to pompt if the chatbot should show information of other metadata of the service, e.g. processing time or which authorities are responsible to offering the service. The user navigates by clicking on the links provided inside the chatbot conversation. The yes/no questions are translated as **intents**, which the users either wants to pursue or ignore.

For a yes answer, the chatbot selects child nodes of a node it is currently pointing to, e.g. a service node and checks if it can show their content if these are not empty. If the user chooses not to pursue the intent, the chatbot skips to the next element (or node) in the loaded list from a JSON file as described above. With the modular structure of the nodes, the chatbot is allowed to traverse each service node and metadata nodes to display the proper information as seen in Figure 3.6.

The content of these nodes is provided through the IT Service Centre Berlin - *IT Dienstleistungszentrum Berlin, ITDZ* and is part of LeiKa. In a similar implementation for the Facebook chatbot Berlina (currently in Beta), Hassmann and Müller [36] explore a similar implementation and conclude with the use of JSONs being a flexible foundation for development.

3.6 Summary

The Berlin use case is complex in the sense that there are many entangled entities and relationships between the hierarchy of the municipalities. Representing these in one hierarchical diagram is counter-productive due to the rich attributes to numerous special cases and the independence of each authority type of the other. LeiKa overcomes this challenged through a system of sectioned serial number for each office. Yet, understanding how each service operates remains a task to the developer to be able to gather an all-inclusive service strategy for a voice assistant. The modular structure of the Solr API is a helpful tool to send queries and get responses to the software solution we seek. A more in-depth look at the content shows potential adjustments required to adopt it to the solution we present. We discover that Berlin.de shows features of REST architecture such as unique identifiers for pages representing entities (locations, services) and pagination capabilities throughout the website (e.g. for appointment bookings).

As such, operating a voice assistant as a new service could be delegated to D115 as an authority under the realm of the German Ministry of Interior with the help of the ITDZ, since both entities have the knowledge base and the content to power the service. As the content is subject to changes at any time, updates have to happen without the user having to update in order not to give customers obsolete information. This is part of D115's policy of assuring correct and helpful information is given. For any software solution we present, the guidelines summarized in Section 3.1 have to be adhered to. Accordingly, special attention has to be given about data collection. We show in our implementation (Chapter 9) how we address this by not storing user sessions in any persistent form. We categorise our system as semi static, since only the content should be interchangeable through an independent API (Solr in our case).

As per D115's guidelines, we should be able to understand from special cases how to maintain a more well-rounded software that handles more specific cases, and not only give general information. As this requires more detail for requirements engineering (tender and performance specifications), we keep this implement for a future stage. This might involve uses of ML to detect patterns and irregularities

Finally, with human interaction being a favourite option for customers, our software solution competes to present an alternative that is persuasive enough to lure customers to use it. For this, making it constantly available to the user with no waiting time on phone queues or in a public office gives a good starting point. And while the software does not have to understand everything the user says, it should be able to abstain from giving wrong answers at the least in order to yield objectively satisfied customers and no frustrations for receiving misleading or confusing information as discussed in Section 1.2.

Chapter 4

The Voice as a User Interface

We start this chapter by juxtaposing the Voice User Interface (VUI) to the Graphical User interface (GUI) respective to cognition and behavioural design which gets us to define new terminological foundation that will follow throughout this thesis.

4.1 GUI vs. VUI

Visuals and Sounds can both communicate the same message even though they use a completely different medium. One one hand, the premise that "a picture is worth a thousand words" can be valid based on the image but only assumes that we want to give a clear picture to the receiver. Communication through voice, on the other hand, can generally allow more room for interpretation. We think of watching a football game on TV vs. listening to the same game on radio as an example, where listening to it allows each person in the audience group to imagine a different game for themselves, even though the match result are the same.

When we use written text, be it in a document or a street sign, or any label, we seek to mostly assure that all readers understand the same message with a universal codification of what we want to express through the language we use. In this process, language becomes the common denominator for understanding and the short written text becomes purposefully designed to give in the best case a message or instruction not open to interpretation and so a foundation for an agreement or contract between source and destination of the message. It is a good medium of reference.

4.2 Utility of Voice

The utility of Voice comes handy with its different paradigm in numerous settings. For a passenger at the airport or an employee between meetings, it can be more effective

to ask questions through conversation than transform them into a query then type them on a screen, or visually navigate on a web page / mobile application until we reach the information we want. In a 2016 Cisco Systems (formerly MindMeld) survey [30], of 1800 users of voice assistants, 61% see that the primary use of a voice assistant is when hands or vision is occupied. Today, this goes further beyond the simple use-cases to become an integral part of our fast-paced life, where we see companies like Ecobee getting around 40 percent of their sales through voice-based AI [4] since their embrace about two years ago. The 10-year-old company's CEO elaborates that their customer "[...] have to fight traffic to get home, and then they have to feed the kids, diaper the baby, and who knows what else. [Ecobee] give them a hands-free way of getting something done while they're in the midst of other tasks" [4] .

Also, in terms of accessibility, blindness could sometimes make voice the only possible way to navigate or be aware of one's surroundings, e.g. a blind person crossing a street depends on the sounds coming from cars and traffic lights. In that sense, voice can allow people to do their activities free-handed and unobstructed by actively engaging into reading activity as is the case when we look on a screen. With our ambitions in connectivity becoming more complex, we increasingly started depending on our phones in the last decade. This is where the use of a Voice User Interface could come at an advantage of liberating us from the constant usage of our smartphones and moving from one screen to another. However, it is noteworthy that the trade-off between visually and auditory information exchange comes at a price. While both have their own advantages and disadvantages, it is important to understand that the context in which the medium is used determines dependent factors.

Hence, we need to differentiate between GUI design and that of a VUI. Since most computers and advanced electronic devices operate with a GUI or a simpler textual interface displayed on a screen, our acquaintance with GUI trump the know-how we developed with VUI. Therefore, it might not be most intuitive to apply the same concepts on VUI design simply since we start from the mindset that the GUI dictates how to use the software and the user has to understand this interface and deal with it as is.

4.3 Why Can't AI Understand Us

AI has made giants leaps in the last few years. Thanks to neural networks, Bayesian approaches in classification, decision trees and many other theories, we are capable of performing rigorous analyses on datasets Considering that speech is one of the most complex forms of data, there are still various challenges that do not have a single strategy to tackle.

> What makes voice-based AI so appealing to consumers is its promise to conform to us, to respond to the way we speak—and think—without

> requiring us to type on a keyboard or screen. That's also what makes it so technically difficult to build. We aren't at all orderly when we talk. Instead, we interrupt ourselves. We let thoughts dangle. We use words, nods, and grunts in odd ways, and we assume that we're making sense even when we aren't. [4]

When speech meets AI, there are a lot of language ambiguities to deal with and multiple contexts to understand. Since this is a field by its own and the amount of problems we can face with understanding natural language is unlimited, it is not the scope of this thesis. We only briefly survey a few categorical examples in the following:

- **Sytanx**: homonyms such as "I *present* you a *present*." or "The *fly* wants to *fly*"
- **Sematics**: metaphors, sarcasm, puns such as "it's raining cats and dogs"
- **Underlying Sentiment**: such as "oh yeah, sounds very exciting" when it's not.
- **Dialects**: enunciation such as [dɪˈvɛ|əpmənt] *(British)* and [dəvˈ|ɑpmənt] *(Indian)*

While Machine Learning enables us to decode phrases not in the classical way in the past by randomly guessing the whole expression, our speech datasets grow and require analysis in further breadth and depth to deliver a sustainable Natural Language Understanding (NLU) engine. So far, in the past six years our primary approach has changed due to the little progress made with it. Instead of trying to understand exact meanings, we work from "imperfect matches at the outset, followed by rapid fine-tuning of provisional guesses" [4]. The learning part is involved by reinterpreting missed expressions [19]. On one hand, neural networks help us understand language patterns and better the more frequently we use them, but can fail in the case of homonyms. [23] Thought vectors, on the other hand, complement them by controlling the connection of different words with their related meanings

Mostly independent from the system we use, it is a rule of thumb that most AI-powered personal assistants like Siri or Alexa are still at an early learning phase. In order to be able to make use of them in the future, we need to traverse a period of immaturity teaching the system. This limits us for instance to currently using short sentences to lessen the probability of failure and the system misunderstanding us. And so, naturally, what distinguishes a good system from a better one is how fast it learns.

Interestingly, a side effect of this indeterministic technique is that the voice assistant inadvertently simulates a person who can sometimes not hear us, asks us to repeat a sentence or explain it with different words. Although this is the same idea that a good system should have adequate incoming and outgoing speech fault tolerance, it is unprecedented to tackle a computer system on the consumer level with the idea that we might very much hit or miss the action we want to perform. On the other hand, though

premature, it is evidence that a human has much more authority to a machine-powered (and maybe autonomous) brain.

Yet, although it might seem at the beginning that moving between both paradigms is easy, the more detailed system design gets, the trickier it becomes to transform GUI elements into text. Particularly with long texts. We will go over this in more detail about voice design guidelines in Section 7.3.

4.4 VUI Semantics

Having the visual or haptic interface disappear is therefore a game changer for as soon as the user does not have direct instructions on how to deal with it, they start to improvise, or rather move away from the idea that the interface constrains them to the illusion that they define that interface and can control it. For instance, the user knows that they have to press only on a certain area of the screen to activate a button and actively move the mouse to that part. In other words, when a user sees a 'save' button on a dialogue box, in their head they relate to it the functionality of a commit and keeping a persistent copy of the current state of the parent file to that dialogue. The label 'save' describes in a visually expressive fashion what the user can or cannot do within this *dialogue state*.

In voice on the contrary, a user is uninhibited in what they can and cannot say and the system is supposed to understand the *intention* of the user with the *formulation* they just used.

In order to translate the aforementioned with less ambiguity, we need to define the lexis we use for voice design utilized in most common chatbot and voice assistants constructs. We divide the building model into *intents*, *utterances* and *slots* The Fulfilment part is taken care of through a back-end and possibly endpoints connected to it containing the programming and business logic to the interface, which also validate the slots before sending it onwards, similar to when JavaScript takes care of checking a box is not empty when we use a form ona website.

4.5 Terminology

We already unveiled in Section 1.3 that Alexa will be the underlying platform. Thus we cap the elementary definitions of voice assistants with distinct Alexa-specific concepts and give meaning to the following terms:

Intent

As the intention the user pursues with a spoken sentence. This translates to the action being executed upon the user's command based on mapping his/her words to this action.

Utterance

As the wording a user picks to express his intent. For instance, to set the volume on a TV to quieter, we could say "turn it down a nudge", "It's too loud" or "lower the volume". Although all three sentences have the same meaning, linguistically, they are fully unrelated and only context makes us understand them, e.g. we have to know that "it" refers to the TV set in close proximity.

Slot

As a variable of a certain type we define in our programme that belongs to a category of items. For instance, when someone says they would like to order a *large* coffee, they assume that there is the option to have a small coffee, too. And so large and small refer both to the size of the coffee. Programmatically, we define `large` and `small` to be of type `size` of the coffee. Similarly, if there is the option to order a tea and a Cappuccino, it would be only fair to define `cappuccino` and `tea` to be of type `beverage`. Consequently, **slots** can be defined to any part of the sentence, where a parameter (here beverage or size) can be grouped into **slot types**.

This idea is to be handled with care, though, since technically in most languages we can define a subject, a predicate and an object. Here is an example in English:

$$\underbrace{I}_{\text{subject}} \cdot \underbrace{would\ like\ to\ have}_{\text{predicate}} \cdot \underbrace{\overset{\text{number of items}}{\boxed{a}} \quad + \overbrace{large}^{\text{size}} + \overbrace{coffee.}^{\text{beverage}}}_{\text{object}}$$

When we design a voice system from scratch, we might have to make it understand how each of these sentence elements can be interchanged with another one of the same slot type. However, since utterances build on they idea of slot types, we can make delegate the system to understand what is important from the sentence through slot types and what other interoperable words are not relevant. In this example, if we say:

$$\overbrace{My\ brother} \cdot \overbrace{wants\ to\ order} \cdot \overbrace{a + tall + coffee.}$$

it is not important to they system to understand at this stage who is going to drink the coffee. It should be more concerned with the intent; making the coffee, e g. sending a signal or instruction to a coffee machine.

Ultimately, it is up to us to draw the line on where we want to define a variable slot, that is relevant to the sentence being said by the user in context at a certain time in the conversation flow and where certain sentences would be as a whole giving the same intent

Synonym

As a word that has the same meaning of another word that fills a slot. For instance, if we assume that a user says "I want a **cup of jolt**" they system should still be able to understand that they want a standard coffee. If, however, the user defines the type of coffee beans they want, such as 'Java' or 'Arabica', the system should still be able to differentiate between these and not consider them both as the plain 'standard coffee', fulfilling a different intent.

Dialogue State

Like with a state machine, when we have a conversation with Alexa, the dialogue goes into states `STARTED`, `IN_PROGRESS` or `COMPLETED` in the `dialogState` property of the JSONs being exchanged with the Alexa client (e.g. an Echo Dot). This is helpful for delegating the dialogue to Alexa or handling it in our own code especially in a multi-turn conversation (see below). More details about this interaction available in the ASK DOCUMENTATION [1]

Dialogue Directives

Taking advantage of the `DialogState` property, dialogue directives are a concept for dialogue management providing a mechanism to conduct dialogues in a multi-turn fashion via Alexa's interaction model directly with little programming effort Without dialog directives, we need to write the logic to collect all the slots (data) we need from users, effectively maintaing the whole session ourselves. As such, dialogue directives are a clean way to manage sessions with one of three options:

- Delegating the collecting of required slots to Alexa
- handling the multi-turn dialogue in the fulfilment code
- using a hybrid of the above.

Entity Resolution

Entity Resolution enables us to use synonyms for slot values without the need to manage this concept in detail in our implementation code. As synonyms are part of the interaction model, resolving them becomes as easy as finding a canonical value in an object (as a complex data type). The concept of entity resolution additionally simplifies verifying that the user indeed said one of the synonyms to a slot value by confirmation

[1] ask/dialog-interface-reference.html#scenario-delegate

utterances from Alexa.
Example: Using entity resolution, we manage to understand *'Perso'* as a synonym to the word *'Personalausweis'* by defining the relationship between both in the interaction model

Fulfilment

Building on the intents that the user gives to Alexa, we need to fulfil the command (after all the prompts and confirmations). This happens in our back end code. Fulfilment is what happens when our back end code runs. The code itself can vary in implementation using different programming languages like Java or Python, in hosting using our own HTTPS endpoint or a Lambda instance, and the result is what the programme delivers. This could range from changing the state of a device, e.g. turning a lamp on to making an API call that delivers text Alexa says. Programme implementation is very flexible to the developer's needs.

Interaction Model

This is the Skill configuration, also considered as a front-end for the Skill. It is effectively a JSON file of a strict schema that contains all information about what the user says in terms of Utterances, Slots and their types (enumerations of Lists which Alexa tries to resolve our spoken text to). From Alexa's side of speaking, the interaction model only contains the prompts Alexa says to collect slots in the dialog management process (see dialog directives 4.5)

Multi-Turn Conversation

A multi-turn (or multi-part) conversation is one, where a dialogue of more than one question and answer takes place. In Alexa's context, this is identified through the `dialogState` property and is responsible for collecting / verifying information before it being sent to the back end for fulfillment.
Example: For the intent `transfer_drivers_license`, which includes a required slot that needs the value `originOfLicense`, which is of type `countryList`, here is a possible

" Alexa, I want to transfer my driver's license "

" What country is your license issued in? "

“ Mozambique ”

“ Okay, <fulfilment_response> ”

Over-answering

This is what happens when the user provides information that could fill a non-required slot. We catch this information by providing all possible utterances a user can say. In the example above, if the `originOfLicense` was not a required slot, Alexa would not prompt us to tell us where the license is issued. If we say in the first utterance to start the intent the name of the country the driver's license is issued in, we over-answer by giving Alexa that information. Depending on our application logic, we can make use of this to make our conversation smarter and not have Alexa ask for information the user already provided, making the voice assistant sound unreliable and primitive. For this, we need to anticipate this effect and include non-required slots in utterances the user might say.

Persistence and Memory

By persistence we mean retention of information that unfolds during a conversation session. As we do not use this concept, we consider our Alexa Skill having no memory, such that it will ask for information it might have asked for before. This is to comply with the D115 guidelines on providing anonymous information and not keeping record or storing any personal data from the user. In general, persistence can happen by storing the session information in a database. For this Amazon recommends the previously discussed DynamoDB (a NoSQL DBMS AWS microservice). Alternatives are open to developers.

4.6 Summary

In short, language is a construct that exceeds the representation of simple expressions in syntax and semantic rules. Conversation management breaks down the complexity of language and is used in most IVR systems. As we move from building grammars based on Chomskyan theories to resolving parts of a sentence to the nearest elicitations using ML, we introduce concepts and tools to do so. Like in any programming language, concepts need to be defined to have a standardised understanding and not be just related to one language (e.g. type inference is a concept and not a feature in JavaScript). The

interaction model of an Alexa Skill handles a generous amount of implementations to these concepts through a compact tree representation. This helps us to delegate the understanding of user's intents to Alexa as a black box and focus our work more on the fulfilment part.

Chapter 5

State of the Art

This chapter evaluates the possibilities we can combine with the previously discussed artefacts of the Virtual Citizen Asistant. Although we do not exclude a communication with web app at the beginning, we exchange the dependency in favour of the underlying API omitting one layer of communication. We start by positioning our expectations from an AI to then compare the Microsoft Bot Framework, Actions on Google to Alexa Skills Kit as potential underlying systems for our final product.

With these introductory definitions we try to lay out a scaffolding to contain a model for a natural language conversation. Hence we make assumptions on our requirements and explore the options that can suit these. We introduce our requirements in the following Section 5.3 after we understand our current standing point.

5.1 Analogies to the Introduction of Web 2.0

Much like when tablets and smartphones were introduced to the market, the languages and framework used to render GUIs to the users had to adapt, which resulted into establishing HTML5 and CSS3 as new standards for the web. In that transition, browsers had to render different versions of the same page or even different pages depending on the client. It started with a fixed amount of screen resolutions that rapidly grew within a few months. Offering a desktop and a mobile version was another transitory solution paving the way to a display agnostic responsive design. Similarly, the MPEG4 H.264 codec accomplished a better adaptability and less battery consumption for video, which were success criteria to toppling the widespread of Adobe Flash on the web. When screen sizes and aspect ratios became also so diverse that it was hard to keep track of, websites adapted in many ways. The processed can be seen as an organic variation and selection.

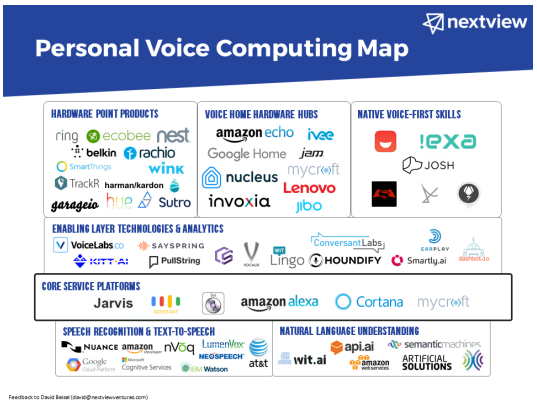
With the introduction of VUI for mass consumption and open development, similar undertakings are expected to happen [7] until we get to acquire an advanced voice user

experience. With diverse APIs available explicitly for VUI design, like Bespoken[1] for performance monitoring and testing of a VUI and Sayspring [2] for voice prototyping, we can utilise a few options which are likely to define new modelling and testing standards for VUI design and implementation. At the same time, we expect a few attempts to disappear in the process. Drawing analogies between RSS feeds and the news feed on all social medial platform, we can put the failure of one standard in perspective to the success of another stemming from the same idea.

As we have seen from previous examples on the web, particularly web searches being the most important, the shift of customers from AltaVista, Lycos, Yahoo! and MSN to Google, Yandex and others shows that marketing, expandability and maintaining quality are key factors to customer retention and survival.

5.2 Development Models and Platforms

Figure 5.1: The VUI-Stack, showing the different layers a voice service can build on. Based on [2]



Chatbots have become available and more accessible to develop through numerous platform (for the same platform (e.g. Facebook), for our own standalone platform(e.g. Lex), or just a tool (Flask)). We no longer have to come up with the whole infrastructure.

This gets us to understand that while it is possible to build a voice service from the ground up, all major platforms already supply the necessities to develop for an existing platform such that a lot of overhead is already implemented. VoiceLabs defines this as the VUI Stack (Figure 5.1).

Further, with each of the voice assistants platform becoming specialised in a different domain, it s not suprising to see synergy effects on the market. Recent news about Amazon and Microsoft agreeing to make their voice assistants talk to each other [33] is one such comparison to how different operating systems today exchange common file formats to do the same tasks differently

[1] https://bespoken.io
[2] https://www.sayspring.com

From End-Users' Point

Before we evaluate platforms in terms of software development, we research the question of evaluating the platform from the customer's point of view. For this we look into empirical data from previous evaluations, then complement it with our own survey.

In his thesis project, Osman [37] surveys Siri, Cortana and Google Assistants with respect to their user-friendliness using Attrakdiff and other procedures. The research quantifies the perception of 24 interviewees (13 women and 11 men between 18-35 years of age) about the voice, the human aspect of the voice assistant, pragmatic, hedonistic and other qualities. It also tests the reactance of the subjects with ratios about knowing vs. using the voice assistant.

Though the research does not recommend a certain platform, it hints that different platform adapt a different characteristic (especially with regard to wit practicality), which trigger different emotions on different users. Already in 2016, the study shows that the names of the voice assistants sound familiar to the sampled subjects.

Other sources indicate that "AI platforms by Google, Apple, Microsoft, and Amazon all show different strengths. Google Assistant is the best on wide-ranging search commands." [4] As Alexa is particularly talented with shopping commands [4], the platform revolves well around sustaining a conversation".

Quoting Microsoft product manager about Alexa, we acknowledge that "Alexa's broader success resides in its ability to alleviate the stresses of an overbooked life. It's the companion that's always ready to engage" [6].

Additionally, there are a few plugins in the making to create a voice profile for Google and Alexa, giving an individual touch of sound based on a profile [8]. This can have a wide-reaching audience, for instance a voice assistant could mimic the sound and language of a person when they are not present. Likewise, a game can have a richer experience with its characters having different sounds.

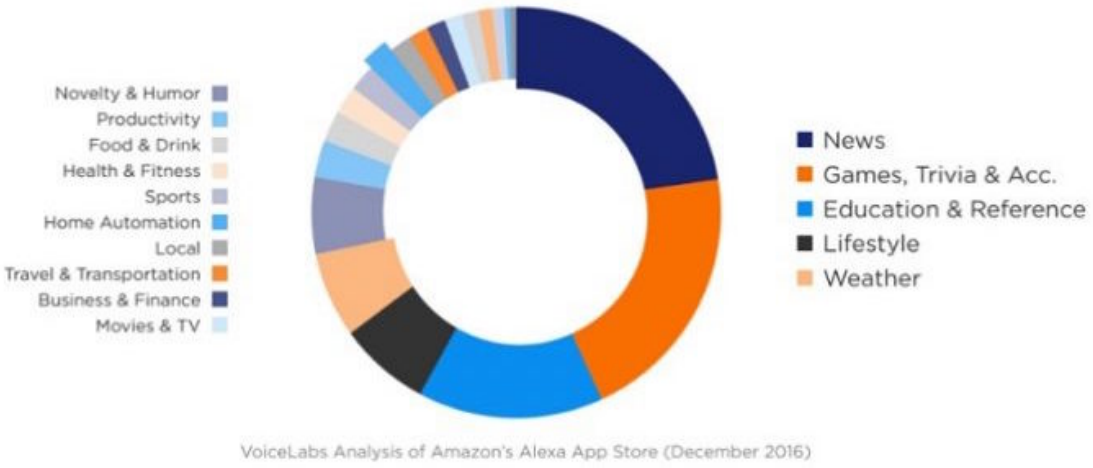
From Developer's Point

The perspective of third-party development focuses mainly on availability of APIs and SDKs. With the exception of Siri being tied to a relatively closed platform and require knowledge of Swift or ObjectiveC, Siri can only be executed if an app is installed on the iOS device. Moreover, a 99 US$ one-time fee to become an active developer able to publish on the AppStore is a prerequisite. Alexa and Google demonstrate more freedom with the option to not need to pay any fee until after the 3rd-party software is in production. This is another reason that gets us attracted to both platforms.

Also, their higher user base promise a higher likelihood for a developed solution to be used on the respective hardware (Android devices, Alexa-enabled devices). Especially with Amazon Skills launching in Germany only since February 2017, we expect that our solution could fill a market gap. Figure 5.2 demonstrates how an Alexa Skill

could cater for an unsaturated market.

Figure 5.2: Alexa Skills by Volume (2016) [1]. Categorised as part of the 'local' section, the percentages demonstrate how an Alexa Skill as a choice could cater for an unsaturated market.



For Google, the option of using Dialogflow[3] (formerly API.ai) was excluded as it was explored in the equivalent project by Hassmann and Müller [36].

With Amazon having an obscure position to data and privacy as opposed to Apple and Siri, this gets us to question how Amazon protects personal data. During the AWS Public Sector Summit 2018 in Brussels, Belgium many answers were provided clearing doubts about the company's compliance to the GDPR and its Cloud Computing Compliance Controls Catalog (C5) attestations as a proof of high standards for different forms of data protection (e.g. resale to partners and third-parties and responsibility on cyberattacks) [43]

Conducted Survey

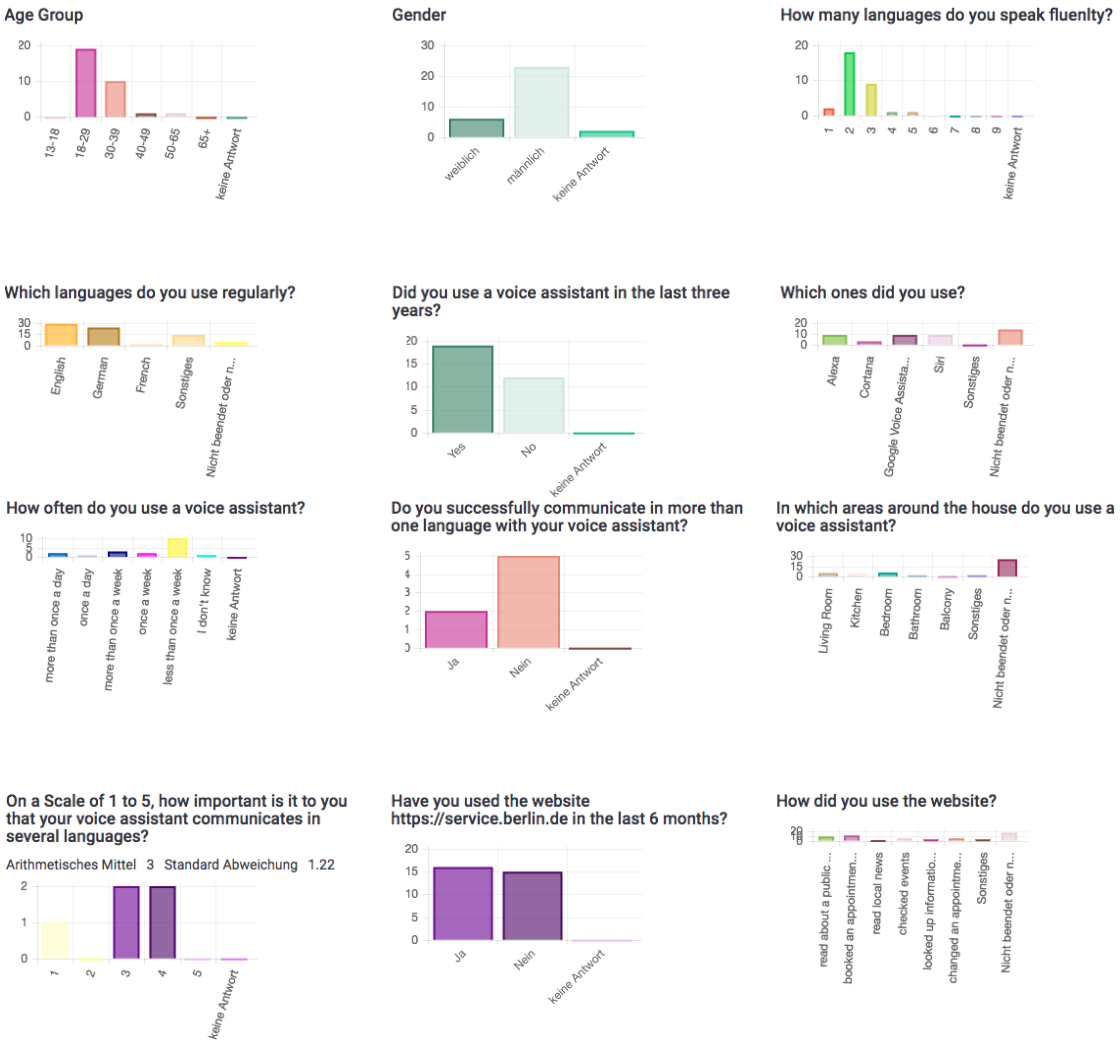
In addition, we survey 34 persons to validate our choice out of the preliminary interest in Alexa.

The survey aims at understanding potential customers' expectations while evaluating their current understanding of the possibility of a voice assistant for the e-Government sector. Titled 'Usability of Voice Assistants', it is divided into the following question sets related to:

[3] https://dialogflow.com/

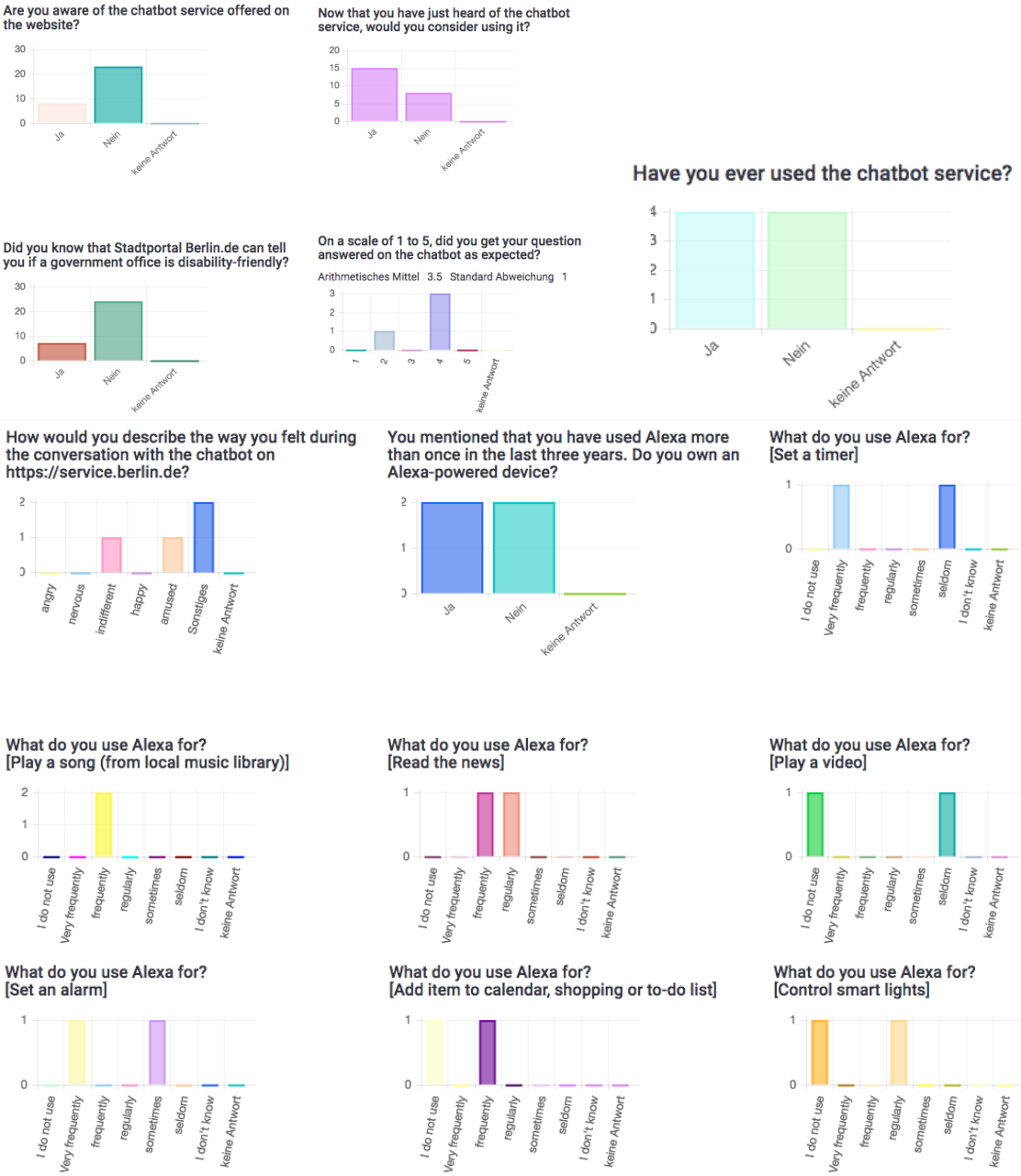
- Demographics
- Berlin.de City Portal
- Voice Assistant for public services
- Linguistic Nuances (Utterances)
- Use of a Voice Assistant
- Alexa as a Voice Assistant
- Alexa Skills

Figure 5.3: DEMOGRAPHICS & USE OF A VOICE ASSISTANT - Results of conducted survey for 34 participants for the development of an Alexa Skill



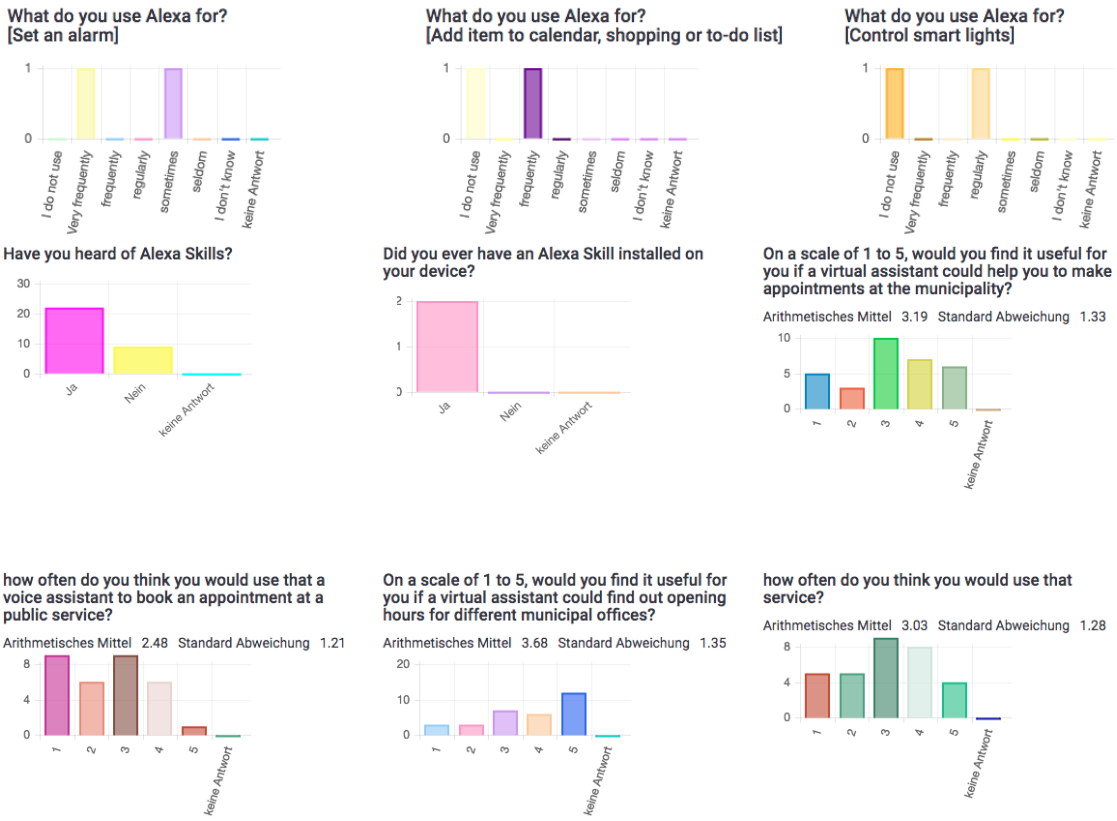
The survey is carried out using LimeSurvey, an open tool (for educational purposes) with extensive analysis tools and the ability to export to statistics software such as R, SPSS and any tool accepting CSV file format. It lays out a foundation for future metrics if used in parts and on larger amounts of interviewees.

Figure 5.4: BERLIN.DE CITY PORTAL, THE VIRTUAL CITIZEN ASSISTANT AND ACTIVITIES ON ALEXA - Results of conducted survey for 34 participants for the development of an Alexa Skill



By combining responses from two or more questions, we find correlations that give us the following key findings:

Figure 5.5: ACTIVITIES ON ALEXA AND A VOICE ASSISTANT FOR THE CITY'S AUTHORITIES - Results of conducted survey for 34 participants for the development of an Alexa Skill



- There is an equal demand for German and English as the VUI language.
- There is an existing user base for the Virtual Citizen Assistant that has potential to grow and extend to a voice assistant.

We therefore decide to build an Alexa Skill after having confirmed that the number of users who have heard of Alexa and are using a VUI are the highest.

5.3 Choice of Platform

After deciding that the voice assistant solution will take the shape of an Alexa Skill, we cover here some structural conditions we anticipate with this decision.

Secure Data Transfer

Using Alexa Skills require a secure HTTPS endpoint. As AWS Lambda is a possible option (as a trusted trigger), we consider it in order not to have to manage any SSL certificates (except for at our API). For Lambda, the first million calls a month are for free. Charges apply per 10ms unit.

Ecosystem

While we can use our own building blocks, AWS offers a good combination with the Alexa Skills Kit, from asset hosting to availing virtual machine instances

Programming Language

Chooing a programming language is relevant, as we have an already implemented template from the Virtual Citizen Assistant which we want to use. At the same time, trying out a new language is a key incentive to start this project. Given that we have the freedom of choice with Alexa Skills between Java, Node.js framework (JavaScript), C# and Python, we look at available resources for each language. Node.js prove to be the most resourceful as will be discussed in Chapter 9. As a secure alternative, the Alexa SDK for Java comes second to Node.js's as opposed to C# which has no SDK for Alexa. With Java and ECMAScript having similar syntax, the arguments towards development on this framework is strengthened.

(Human) Language

based on an excerpt from ASK documentation: Availability [4]

- [A Skill hosted on the German store only with an interaction model in locales 'de_DE' and 'en_UK' can be made] available [.. with the] selected country distribution [before deployment].
- [It] can be used by customers in Germany who have set their devices to use German.
- [It] can be used by customers in Germany who have set their devices to use English (UK).
- [It] cannot be used by customers in Germany who have set their devices to use English (US) or any other language.

[4] ask/develop-skills-in-multiple-languages.html

- [It] cannot be used by customers in any other country, regardless of the language they've selected for their devices.

We understand the choice of platform is significant at an initial stage, since it decides on the architecture of the software we develop with. Given that we will have a constraint related to the number of users, we find through the conducted survey that Alexa Skills Kit and AWS are worth embracing as a bilateral ecosystem, to users and developers. In the next chapter, we familiarise ourselves with potential and limitations of this choice.

Chapter 6

Amazon's Ecosystem

After having a sneek peak into Alexa Skills,we proceed by comprehending the backbone that operates it. In this chapter we discuss Amazon's state-of-the-art strategy as it is important to introduce the implementation scope of voice assistants top down prior to exploring the current context after comparing it to other approaches in the larger context of conversational bots as a whole from a technical and user experience point of view.

6.1 Amazon Web Services (AWS) + Alexa

Getting started with Alexa as a platform for the first time might seem a little overwhelming especially since each component of it is being constantly restructured since its debut release in November 2014 and with Q4 2016 being the beginning of its major market penetration success [15]. As throughout the course of this thesis these major changes occurred, we discover that Amazon's philosophy and success is based on the divide and conquer principle on a large scale. This allows loose coupling of all sorts of components of the company, starting from their business logic to even their branding themes.

AWS

Amazon Web Services is a Cloud Computing service provider with a complete set of services to help build and run web applications "reliably and securely at a cost and scale" according to one's independent needs [38]. It comes with agile abilities to adjust to various solution at a flexible scheme with benefits like multiple server farms globally (operating under different legal contracts with respect to data security and user privacy), caching, NoSQL, DevOps and so forth.

Alexa + Skills

Alexa is a cloud-based voice service assistant platform by Amazon powering millions of devices on its own-branded IoT devices like the Echo, Echo Dot, Tap, FireTV as well as cross-platform through mobile apps available through Apple's AppStore for iOS and Google Play Store for Android devices. Unlike Apple's approach with Siri for instance [1], where including support for third-party integration on iOS's Service Development Kit (SDK), Amazon decided with the launch of Alexa to include non-Amazon developers right from the start by introducing a multitude of (SDKs) around the platform as part of AWS, e.g. Alexa Skills Kit SDK discussed further below.

So far, though the separation of AWS and Alexa's environments is not linguistically intuitive with the company's name labelled on every service component [2], Amazon.com, Inc. offers an array of web services through AWS that summarize all building blocks necessary to operate Alexa as a software for end-users. Of course with the introduction of the aforementioned devices, it becomes intuitive to call these 'Alexa devices' since their primary purpose is to operate as Alexa clients. We refer to Alexa here only as the service offered by Amazon to consumers. Consequently, although Alexa comes as a fully packaged service to end-users, we can reduce it to a compilation of many micro-services provided by AWS. As most of these are available separately in the form of Software as a Service (Saas), we conclude that AWS incorporates the following components required to make Alexa come together and becomes a consumer of its own web services platform.

6.2 AWS Modules as Alexa's Building Blocks

listed in sequential order of importance, Alexa's service modules include but are not limited to:

Lex [3]

for conversational interfaces using Natural Language Understanding, Text-to-Speech and Speech-to-Text

"a service for building conversational interfaces into any application using voice and text" [38]. Lex is the most important backbone to make Alexa possible and can be a main operator of another software package to create a whole new category of independent voice assistants and conversational bots. Independence denotes that from a VUI

[1] as know from its policy on new software and hardware products like the case with the iPhone, the Mac and its other product lines

[2] see Etymology in Annex 11.3

[3] `https://aws.amazon.com/lex`

Stack like Alexa's with the Echo devices and respective platform, software (Skills and APIs) built on top of it, or with Siri, where the Hardware is the iOS-operating device and the software being an extension of the mobile apps using Apple's SDK. It would also mean adopting the Alexa API to a new kind of hardware, such as cars, which goes beyond the scope of this work. And while this independence means more flexibility in developing a solution of our own, this would often mean a much higher effort since we miss the opportunity of taking advantage of the already built and tested artefacts. The most critical argument against development with Lex directly is that it was not available in German at the time of this research. We therefore decide to use the Alexa Skills Kit with Alexa as a product and not as API, which ultimately takes advantage of Lex and the other components described below under the hood while still remaining in a familiar and commonly used environment. Lex and Alexa use the same deep learning techniques for natural language processing with the workflow described with the example in Figure 6.1 below. Lex's difference to Alexa is discussed in Appendix 11.3.

Polly [4]

for speech synthesis

another "service turn[ing] text into lifelike speech, allowing [developers] to create applications that talk" [38] wile harnessing the power of deep learning. It is more or less the mouthpiece of Alexa built on top of speech synthesis algorithms.

Transcribe [5]

for automatic speech recognition using Speech-to-Text

which "makes it easy for developers to add speech-to-text capability to [...] applications" [38]. With Transcribe we are able to get text out of the user's voice before passing it into a format Alexa's backend would understand.

Lambda [6]

for intent fulfilment

Although Lambda is a versatile "service [built] for a variety of real-time serverless data processing systems" [38], it can be used as an integrated server instance to host back-end code for intent fulfilment.
Pricing: First Million calls are free. Billing charges per operation millisecond.

[4] `https://aws.amazon.com/polly`
[5] `https://aws.amazon.com/transcribe`
[6] `https://aws.amazon.com/lambda`

DynamoDB [7]

for persistence storage
Acting as a very fast NoSQL database service while also using tables, Dynamo is very scalable and can be primarily used in the context of Alexa as a persistence holder for data collected during the interaction with the user to give Alexa a 'memory'.

S3 [8]

for resource storage
standing for simple storage service, it is a safe instance for cloud storage using buckets (namespaces) with various access management and encryption options.
Pricing: The first 5 GB can be stored for free.

CloudWatch [9]

for event logging
CloudWatch acts like the console for an operating system (OS). It monitors all low-level events happening within the AWS sphere. In combination with Lambda it comes as handy tool to log events resulting at runtime once a Lambda instance is called and its code being executed and is used for debugging.

IAM [10]

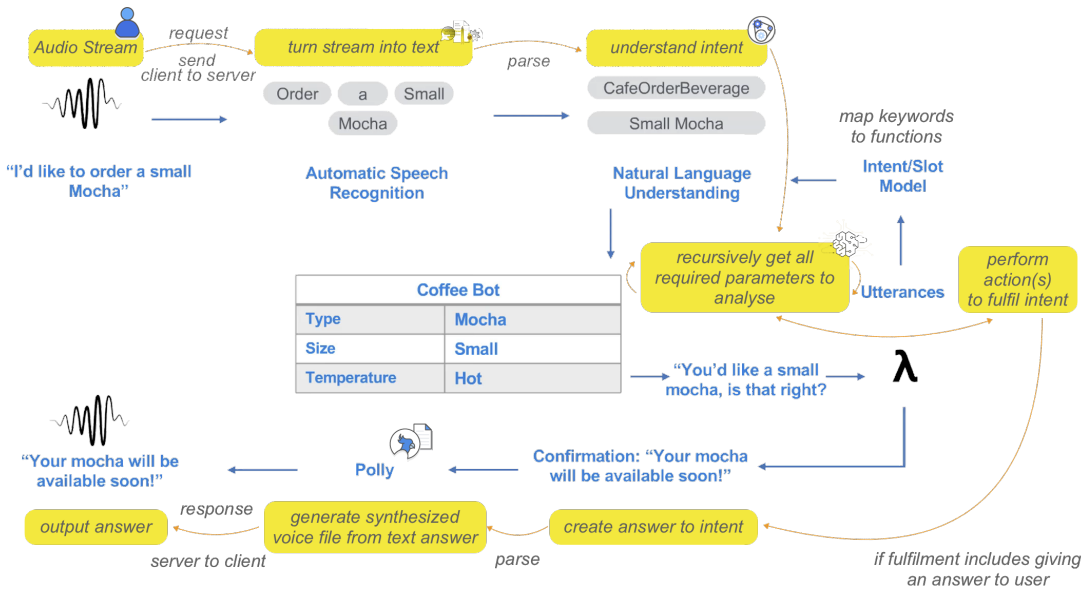
for identity access management within AWS
Identity Access Management (IAM) is a secondary service module regulating in the Alexa context in combination with Lambda the routing rights between internal Amazon endpoints. It also ensures compliance policies are enforced within and between AWS modules, as well as between AWS modules and other external services.

Cognito [11]

for identity access management beyond AWS
like the rest of AWS's modules, Cognito is a service to perform user authentication and can be used in combination with an external endpoint instead of Lambda.

[7] https://aws.amazon.com/dynamodb
[8] https://aws.amazon.com/s3
[9] https://aws.amazon.com/cloudwatch
[10] https://aws.amazon.com/iam
[11] https://aws.amazon.com/cognito

Figure 6.1: Interaction between AWS modules in the use case of a “Coffee Bot” based on Niranjan [19]



Putting different combinations of these and other building blocks interactively together generates the model for Alexa. In a world increasingly operated by Internet of Things (IoT), we describe a possible interaction in the example here for a use case of a hypothetical chatbot that operates a coffee machine using some of these service modules (Figure 6.1). Although this graphic describes how an end user would combine these modules to set up their own chatbot, this is the same workflow that Alexa uses. Hence, with Amazon’s use of its own micro-services we infer that takes advantage of these to build a whole new ecosystem putting Alexa Skill developers as producers, end-users as consumers, and the Amazon website and Alexa App as a marketplace to mediate between the products (Skills) the developers produce to their target customers (end-users, country-specific or worldwide). From a marketing perspective Amazon achieves through Alexa a vertical diversification of its product programme (where existing AWS services result in a new product expanding the value chain) while simultaneously offering an extension of its aggregator model (as opposed to a marketplace model)by playing as a mediator between the developer’s role and that of the end-user’s

we can therefore describe the meta-model for Alexa similar as one of an application with several front-end and back-end components acting in a DevOps environment of cloud micro-services. Unlike in most GUI-based scenarios with an MVC design pattern [10], where the user uses the `controller` to manipulate the `model`, which in turn updates the `view` appearing to the user, in a VUI scenario, we need to consider

that the user's paradigm to the `view` component is quite different. Before we dive into the VUI paradigm, we introduce Alexa's own implementation of it for third-party applications (i.e. Skills). These are broken down into the following elements for users and developers:

Alexa Skills Store

[for End-Users] where someone with an Alexa-enabled device can preview a Skill before installing it, Once installed, the own instance of Alexa becomes "smarter" by that Skill, which does not need to update from "client side", since it is only linked to the Amazon account and not hosted on the client (only sends requests through it). This offloads the user from the overhead of thinking about updates since these happen in the back-end.

Alexa Skills Kit

[for Developers] Although it is hard to define it as a complete SDK for Alexa and it is still in a continuous expansion phase, it is responsible for compiling the loosely coupled tools provided by AWS and others to act as an interface for the skill from a developer point of view. This fits into the rest of Amazon's scheme of focusing on interoperable micro-services that fit multiple purposes. It includes the following essentials:

Alexa (Developer) Console

This is where all the developed Skills live. It is the gateway to the front-end of the Skill or Voice service (described below). It provides a web interface for initial setup of the Skill, a structured representation of the JSON Files that include the LANGUAGE (INTERACTION) MODELS, the SKILL PROPERTIES, the endpoints it uses and accounts it links to. Throughout this thesis, the console has undergone major interface upgrades and additions to its core functionality. Currently it is also the place where submitting the skill for publication happens and where the simulator (discussed below) lives.

ASK CLI

A Command-Line Interface tool that interacts with the Alexa Console skipping the web browser. Although not at a stable release yet, it has numerous functions to creating, deploying and testing our Skills. Available from the Node Package Manager (NPM) [12]

[12] through `$ npm install ask-cli`

Alexa Voice Service

A service for accessing cloud-based Alexa capabilities with the support of AVS APIs, hardware kits, software tools, and documentation. Through the Alexa Voice Service Amazon has simplified the creation of conversational interfaces for device makers, allowing developers to add Alexa and intelligent voice control to new products for mobile phones and cars to smart speakers and home appliances. It simplifies building voice-forward products by handling complex speech recognition and natural language understanding in the cloud [13]

By exploring these products, building blocks, platforms and runtime environments, we have a deeper understanding of how to make use of these in our implementation and which know which tools to prioritise for development and testing. With microservices, the vast options Amazon offers through AWS and the Alexa Skills kit provide a powerful foundation to support DevOps.

6.3 Skill Structure

After presenting the core stones that take part in administering the Skill, we move on to the Skill structure. Ultimately, the elements discussed above are represented in JSON files that together with the back-end of the skill make up this web-app like composition. Because of its flexible file format, Skills back-ends can be built using one of multiple runtime environments including Node.js, Java, Python, C# and Go. Not coincidently, these are the same languages supported by AWS Lambda

Disregarding the programming language used for development, these statically generated files are included in the code:

`skill.json`

The Skill manifest file including its publication name on the Amazon website, invocation name in each language, a description of the Skill readable by a human, author of the Skill and other properties ⌈ include our sample in Appendix? ⌋

`<locale_code>.json`

The interaction model of the respective language. It is named after the locale the file is in and includes all slots, utterances, dialogues, custom defined slot types inter alia (file structure in Table 7.1).

[13] `https://developer.amazon.com/alexa-voice-service`

`<packageFile>`

A manifest file for required dependencies (installable through NPM for instance in the case of Node or Maven in the case of Java) for installation to run the web-app in Amazon's cloud.

6.4 Alexa in the Eye of the Beholder - Interfaces and Operation

Briefly going over the options available to use Alexa, we present and compare the different Hardware product lines and software solutions developed by Amazon or compatible with their voice assistant in terms of usability configurations, i.e. presentation of voice and graphical interfaces.

Hardware Interfaces

The following devices are Alexa-enabled out of the box. The voice service is either accessible through voice command within the same room or an active button press on the device.

Table 6.1: Currently Supported Alexa-Enabled Devices in Comparison

	Speaker	Tablet	SmartHome	TV
Models	Tap, Echo - Dot, - Plus	Echo Show Kindle Fire	Echo Spot	FireTV Stick
Screen	No	7.0"	2.5" round	HDMI Display
Line Out	Yes	Show: Bluetooth Kindle: Yes	Yes	via HDMI
Alexa On	Voice Command	excl. Fire HD 10 Button Press	Voice Command	Button Press

Software Interfaces

while the hardware models stated above are possibilities for testing, too, they do not always serve as a primary testing environment. There are sometimes more optimised

ways to automate Skill testing, for instance by running scripts that would send the transcribed text in the appropriate JSON format. This is helpful for multi-turn conversations and retaining sessions, so that one does not need to repeat the full conversation until one reaches the test breakpoint.

Alexa App

designed to be a control unit that operates most Alexa-enabled devices, is not an Alexa interface[14]. However, apart from being a tool apart from Amazon's website to install Skills and manage accounts linked to Amazon (for music streaming and other related content-based services), it is a very useful tool to track the history of conversation that took place through a respective Amazon account. We use it mainly to hear what we said and see how the text was interpreted using Lex's NLU engine. It helps for checking homonyms and nuanced utterances.

EchoSim.io

"a browser-based online community tool for developers that simulates the look and feel of an Amazon Echo" [15].

Reverb App

An iOS / Android app, allowing the interaction with the Alexa instance linked to an Amazon account. Perfect option to make a qualifying device Alexa-enabled [16].

Alexa Simulator

Alexa's own web-based online simulator giving JSON responses, card renderings and voice feedback to the requests sent and is part of ASK. We interact with it either through voice or with JSON requests. The JSONs it produces can be used as input for other tests, too.

CLI Simulator

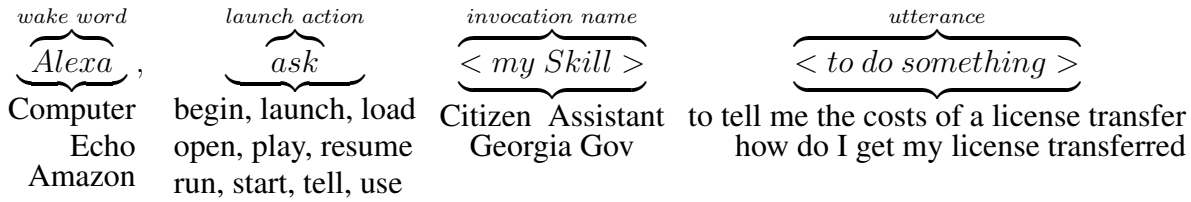
the command-line tool of the aforementioned simulator. Accepts strings and file uploads from the CLI and returns responses to the same interface. Obviously because the Skill is a web service in the cloud, the CLI also requires an internet connection [17].

[14] yet (although this will change soon)
[15] `https://echosim.io/`
[16] `https://reverb.ai`
[17] running command `$ ask simulate --text <inputText> --locale <inputLocale>`

Speaking with Alexa

We end this chapter by explaining how Alexa is designed to operate from the user's point of view particularly for Skills. A full guide is available in the ASK DOCUMENTATION in multiple languages. [18]

A conversation starts with any of the four wake words below, followed by the launch request, then the invocation name and lastly the utterance to action. Here is an example in English.:



6.5 Summary

With this insight we are able to test our Alexa Skill through various methods, depending on the test type. Tests should be run individually and combined. We also have a separation of concerns between development and tests for the interaction model as well as the Skill back end code.

[18] ask/understanding-how-users-invoke-custom-skills.html

Chapter 7

Skill Design

✍ **TO-DO**: *moved from availability section in choice of platform:*
We choose to offer our skill only in Germany because our Skill's functionality is specific only within this geographical region. For that we configure the Skill to handle English (US) and German.
Consequently we will two interaction models and two handlers.
say why we do it like this 3ashan each language group has a different interest although we can do a handler function for two languages but we prefer not to 3ashan ne3melha schöne und nicht systematische implementierung.

✍ **TO-DO**: intro on structure of chapter
- to then elaborate on implementation requirements in section 7.1. - on Skill design: nobody cares abt ur skill - functional requirements (although this is not OCL now, but requiring an SSL cert.) / non-functional (bot muss höflich sein)

we say first decision was to go for alexa so we see what we kind of frameworks we need around it

✍ **TO-DO**: then talk about how you researched on alexa's available skills and found out that the e-Government sector is underrepresented and hence you chose this as ur analyzed scenario.
fluidly contiuning from intro and background:
- since among Apple and Google it hast the voice-first devices best equipped for its platform, a user base larger then its competition and provides the most mature API and SDKs.

- We choose this skill scenario since it is underrepresented in that pie chart. ...
- amazon as a platform is compared to apple and google readiest (check ref voicelabs)

Table 7.1: Interaction Model Schema (Skill Management API) [12]

Field	Type	Description	Required
`languageModel`	object	Conversational primitives for the skill	yes
`__invocationName`	object	Invocation name of the skill	yes
`__intents`	array	Intents and their slots	yes
`____slots` (Intent Slots)	array	List of slots within the intent	no
`__types`	array	Custom slot types	no
`__samples`	array	Sample utterances for the intent	no
dialog			
elicitationRequired			
elicitation			
type (slot type)			
name (slot name)			
prompts			
__ id			
____variations			
______type			
______value			

✍ **TO-DO**: It's important to say that they do not transcribe, but do term weighting etc, a black box, and that we will not hear everything right, just as we as humans do not

✍ **TO-DO**: - ..it would speak as an advantage for bots if they can determine these things automatically..
- currently most tasks revolve around performing tasks like setting an alarm,
- answer suggestions functionality in chatbot equivalent - next step is to get around the user's frustration by making the bot at least more human.
- Alexa Skill will work in Germany in english and german -¿ add english after german

✍ **TO-DO**: MAGENTA
-AL: Ich würde erst etwas die Algorithmen und Datenstrukturen (Textanalyse, JSON, ggf. Graphen beschreiben).
-AL: Anschließend die Frameworks vorstellen
-AL: Wichtig ist: Aus den Beschreibungen eine Schlussfolgerung ableiten, welche Art von Lösung entwickelt werden soll.
for current bot:
- Lucene **as the golden standard**: spell check, unscharfe suche, Tika / detect language / ...
- Solr - explain what's an intent, whats a slot `https://service.berlin.de/virtueller-assistent/virtueller-assistent-606279.php` `https://www.itdz-berlin.de/`

7.1 Frameworks and Data Structures

Node.js

Out of the languages supported by ASK and Lambda, we decide to use Node.js, a JavaScript (ECMAScript 6) framework due to its event-driven nature and to take advantage of its non-blocking I/O model. Being single-threaded, Node.js guarantees high performance at large scale with large volumes of requests considered. With its JavaScript (ECMAScript) foundation, no wonder it is becoming a standard in web-apps. Hence, the decision also comes due to the richness of develpers' experience with the implementation for Alexa Skills.

The concept behind modularisation in Node.js is to keep Node's core small, efficient and tight in terms of functionality it provides. There are three kind of modules:

- Built-in modules that are part of Node core. These are mostly for essential things like reading and writing into the file system, making requests.
- third-party modules. Up for development by the community
- local modules

✍ **TO-DO**: below

- start with saying that the other group doing the facebook bot explored a bit on python with flask, so we wanted to enrich the knowledge base (we mention this in **additional API** and **Choice of platform**)
- Server-side, browser side (Chrome V8), App layer, data layer

- because it can read our JSON easily and fast
- talk about Methodenaufbau (syntax) and firing events
- Event driven like listener-observer model, emit
- list emits table here: https://github.com/alexa/alexa-skills-kit-sdk-for-nodejs/blob/master/Readme.md#response-vs-responsebuilder
- explain directives, we use dialog directives like this: ask/dialog-interface-reference.html#delegate

Apache Solr

✍ **TO-DO**: show each core structure. you mention this in the implementation section

w lucene w tika w nutch wel habal da if required

the API endpoint was originally created for the Virtual Citizen Assistant, likely through a crawler (e.g. Apache Nutch) that parses the information from the Berlin city portal and places it into the respective nodes. The latest information we obtained about the chatbot system entails that the API updates its information periodically, based on the services offered on the Berlin website. ...as a web app and appears to be built from information offered on the Berlin.de websites. The information is organised in 6 cores, each containing information related to different content the chatbot can provide. These are listed below. More information about each core's content is discussed in Section 7.1.

using state of the art standards in TF/IDF for seach queries - fields

-Vorgehensweise: XML/JSON //- index über Lucene //- SolR Knoten...based on sth like when i say "am 10. august' it gets me masalan events..aha august ist ein monat, monat relates to calendar, calendar relates to events

Alternatives

Using Java

Setup in Java using Maven [1]

```
<dependency>
<groupId>com.amazon.alexa</groupId>
<artifactId>ask-sdk</artifactId>
<version>2.0.2</version>
</dependency>
```

[1] https://github.com/alexa/alexa-skills-kit-sdk-for-java/wiki/Setting-Up-The-ASK-SDK

ASK SDK for Java[2]

In this example service writes information to a log and then calls a method to return a welcome response. handle-requests-sent-by-alexa.html#launchrequest

```
@Override
public SpeechletResponse onLaunch(final LaunchRequest request, final
 ↪  Session session)
throws SpeechletException {

  log.info("onLaunch requestId={}, sessionId={}",
   ↪  request.getRequestId(),
  session.getSessionId());
  return getWelcomeResponse();
}
```

Using another HTTPS endpoint:

When deploying your service as an AWS Lambda function on AWS Lambda (a service offering by Amazon Web Services), you configure the function with a handler. You can create this handler by extending the SpeechletRequestStreamHandler class provided in the library. The handler then dispatches requests by calling the appropriate Speechlet methods (onLaunch(), onIntent(), and so on). When deploying your service as a web service on a cloud provider, you extend the SpeechletServlet class. This class is an implementation of a Java EE servlet that handles serializing and deserializing the body of the HTTP request and calls the appropriate Speechlet methods based on the request (onLaunch(), onIntent(), and so on).

Additional APIs

✍ **TO-DO**: - Sayspring: helps developers prototype and build the voice interfaces for their Amazon Alexa and Google Assistant apps.

- swagger

- Flask

-oauth for acct liking https://developer.amazon.com/docs/custom-skills/speech-synthesis-markup-language-ssml-reference.html
- mp3 encoding ffmpeg -i ¡input-file¿ -ac 2 -codec:a libmp3lame -b:a 48k -ar 16000 ¡output-file.mp3¿
for ssml builder npm lib to avoid &amp etc https://npm.runkit.com/ssml-builder

answethepublic.com

[2] https://github.com/alexa/alexa-skills-kit-sdk-for-java

✍ **To-do**: MAGENTA
- as an example for voice
-System Specifications
-System Structure
-UML Diagrams
-Design Choices
-scopes and granularity

https://en.wikipedia.org/wiki/Amazon_Alexa https://medium.com/@robinjewsbury/how-to-create-bots-and-skills-for-facebook-messenger-and-amazon-echo-4a03935eeca1 - Alexa Appstore had over 5,000 functions ("skills") available for users to download,[18] up from 1,000 functions in June 2016. McLaughlin, Kevin (16 November 2016). "Bezos Ordered Alexa App Push"Paid subscription required. The Information. Retrieved 20 November 2016.

Perez, Sarah (3 June 2016). "Amazon Alexa now has over 1,000 Functions, up from 135 in January". TechCrunch. Retrieved 5 August 2016.

- swagger for handling JSON requests?
- https://github.com/alexa/alexa-skills-kit-sdk-for-nodejs

Description of the Interaction

✍ **To-do**: seqence diagram as found in -inbox-

7.2 Challenges

- und Lösungen dafür
- eine Überführung in Alexa, not writing everything new in alexa. such that when you want to do it in another system what do u want to integrate?
- use external web service maybe? in case that helps instead of alexa doing everything..
- konten hosting to be on alexa
- Ähnlichkeitsmaße -levenstein-distanz, IFTTT

ending an utterance with a Fragezeichen Error: There was a problem with your request: "werden?" in the sample utterance "TestIntent was soll aus dieser Skill werden?" is invalid. Sample utterances can consist of only unicode characters, spaces, periods for abbreviations, underscores, possessive apostrophes, and hyphens.

do not use "?"

problems with countrylist cote d'ivoir, südsee (british territories overseas, ...)

7.3 Design Guidelines and Documentations

✍ **TO-DO**: list all guidelines and docus here
Memory (Session, Context)
Entity Resolution
Interaction Model

Chapter 8

Design Best Practices

Throughout the model building process, we come across various nuances and details that might look subtle from a programming point of view. These can enrich or heavily obstruct the user experience, resulting in the users not returning to use our Skill (or chatbot / voice assistant in general) and consequently result in 1-Star reviews technically killing the product.

In this chapter we survey best practices to maintain when building a chatbot or a voice assistant. In some cases it is necessary to separate the former from the latter to obtain better results.

8.1 For Voice Assistants in the Use Case of Alexa

It's important to understand and acknowledge that we are at a turning point in VUI design.with the fast progress made, the design process is still not streamlined. This might sometimes require setting our expectations a bit low, as side effects can occur sinceperformance of an AI today is hardly optimised to understand context like a human would.

Based on our own exploration and through a growing voice design guide offered to Alexa developers [1] as well as personal recommendation from Alexa evangelist Memo Döring [29], we share a few good practices to consider before starting and throughout the development process. These are concerned partly with building the interaction model, as well as with the fulfilment back-end. Many of them also apply to voice assistants other than Alexa

[1] `https://alexa.design/guide`
`https://developer.amazon.com/designing-for-voice/what-alexa-says`
`https://developer.amazon.com/designing-for-voice//design-process/`
`https://developer.amazon.com/designing-for-voice/voice-design-checklist`

Design before Implementation

Since the Skill code usually starts small, it might be tempting to work in iterations, develop for one scenario and then think of what else is needed as we go. As with any complex piece of software, it is very important to have a clear focus on the sequence of our workflow. Design takes an important role in this since it is the base upon which we build our Skill. If we have a bad design, we will end up getting stuck during the implementation. Good voice design means writing down full conversation flows, sentence variations, highlighting order of words, eliminating unnecessary utterances that could result in overfitting, letting Alexa deal with the dialogue management instead of using too many utterances.

In the code part, it might be useful to use routers, intercepts, or both. Routers can be thought of as 'super-handlers' that route our words to an intent, no matter what we say in whichever order. They can be thought of as keyword-sensitive listeners. Intercepts can be thought of blocks as code that would perform before and/or after a certain router takes us to a certain intent.

The initial checklist can grow quickly and immensely but can also become very theoretical. It is therefore for every developer to figure out how to adapt to the VUI paradigm based on the fields of implementation and languages relevant to them.

Natural Language Conversation

When we are prompting for values, they are in a canonical order for how words should be structured when we have multiple adjectives. For instance in English we would say:

$$the \overbrace{big}^{\text{size}} \cdot \underbrace{brown}_{\text{colour}} \; bear$$

because saying "big" and "brown" in the reverse order would not make much sense. Similarly, there is the rule of place before time, which results in sentences like:

$$\overbrace{every\ Saturday}^{\text{time}} \cdot I\ go\ to \cdot \underbrace{univerysity}_{\text{place}} \cdot \quad \text{or} \quad I\ go\ to \underbrace{univerysity}_{\text{place}} \cdot \overbrace{every\ Saturday}^{\text{time}}$$

Many other examples are not actual rules, but more de facto in the language, so we would use them because of our linguistic logic justifies it as a sound pattern, which eventually make us speak a language. There are many rules regarding shapes and abstract concepts. When we build an conversation script (interaction model) programmatically, it is possible that one would mix up in the order of words in a sentence or even in the order of sentences. Since this can result for instance in sentences that are too close to

each other in different intents, this would confuse the system and should be avoided to keep intent groups apart.

Since this changes in every language, it is important not to take one language model and just fit it into another language, since there is much more that goes into that. For instance, it would not be very common to take a 'en-US' language model and fit it into a French one, where the temperature scales or other units are made for another locale.

Question wording

There is nothing more important a Skill can do, than asking the right questions. This does not only concern the formulation of the question, but also the purpose of the question. E.g. it would not be necessary to get a user's geolocation, if they ask for the weather, since the postal code is enough information to take from them as weather would not change dramatically within a district or an area. Collecting GPS coordinates from an Echo Device for instance is not an option, since these are not equipped with one. So, if we need approximate location, asking for precise location is bad practice.

Further, asking for long numbers, such as Model / Serial numbers is not good either, since the user will very rarely get that right. A Skill for tracking packages, where the user needs to spell out the tracking number is likely to fail and the user might just go and track it in a browser. This translates immediately into a loss since they will no longer use the Skill. We need to make speech happen without pauses so that the system understands it as one sentence altogether. Given that there is a lot more happening under the hood than transcription, Alexa works better with words and sentences than one-off numbers and letters.

Context is King

Alexa provides many tools for handling context, such as session attributes. This makes invoking a Skill for the first time be handled differently than the second time the same Skill is invoked. For instance if we say:

User

"I want to bake a cake"

Alexa should respond with something like

Alexa

"Okay, you need {ingredient1}, {ingredient2}, ..."

then when we get back to Alexa an hour later and resume the conversation with a 'resume' request (simply by saying "resume `Skill_Name`), it should retain the context from last time to continue asking after invocation something like

"Did you get the ingredients?"

and proceed to the recipe only once the user answers with a

"yes"

and not restart the Skill from the beginning. In the meantime, we are not listening to the user. Just as it is embarrassing to a person to forget someone's name right after they meet them, not retaining the context of when the user was last seen using the Skill can make it sound dull and a bad way to communicate since it gives the impression of not being reliable. A user could think if Alexa cannot retain the simplest information, they would not trust doing more complicated things with it.

Maintaining context is important between sessions is just as important as across multiple sessions. For instance, greeting the user the first few times, should not sound like the same greeting after a month of daily usage. The user does not want to know every time how to use the skill for instance. Removing an extended greeting message saves the user time.

Moreover, retaining session should also be considered within an intent, such that if we say

"Did Tom Hanks win the Oscars this year?"

"No."

and then follow with a question asking

"What about Katja Benrath?"

Alexa should still understand that we are asking within the '`OscarsIntent`' and differentiate between an '`actor`' slot and a '`director`' slot to match it within the same intent to the right query response. This is only doable by retaining context.

Localisation

Before we avail our skill to another country, it is important to check that the other country does not already have its own skill before we do double the amount of work. And as discussed above, things like linguistic and local changes beyond the classical imperial/metric systems (e.g. shoe sizes) could come into account.

Asking for permission

As mentioned above about taking the least necessary information to give the user the right answer, we want to manage a good balance on the trade-off between security on usability. If we ask the user to make complicated passwords and use tokens etc, no one will use our software. If we make the software too easily accessible, we might jeopardise its success for security problems. Same applies to the downside between functionality to levering specificity. As we do not want Alexa to spill a secret, we would not want to have someone in the room walking accidentally past hearing Alexa say how much money is in our bank account. So Alexa should not in that case start the Skill with your account balance in the greeting message. We also do not want our neighbour next door be able to unlock our car with a generic command or one they may overhear.

One Breath

If we cannot speak out a sentence Alexa can say in one breath, it is too long and should be broken down. Same goes for what Alexa would expect as input. So we should speak out a list in one go, but instead list up to three items and prompt the use to ask for more if they want. Context in that respect is also important. If our Skill wants to list public offices nearby, it does not make sense sometimes to mention the office if it is not open by the time we would get there for instance. That's why APIs have to be studied well before they are just linked to a Skill unlike the case with GUI where the user can scroll

Opposite to the GUI paradigm, more is not necessarily better.

Relevance and Repetition

We want the user to know with subtle hints that we are talking in the Skill about the same thing without repeating our text over and over. Implicit confirmations are good practices sometimes and bad at other times depending on the time and the focus we need to give out to hear a certain sentence. For instance for a purchase, we want to hear it in a full sentence. For just a slot confirmation for instance it may be better to repeat it in the next sentence.
Example:

"What time does today's Lufthansa flight arrive from Cairo?"

"LH 583 arrives to Cairo today at 7:40 PM "

and not just

"7:40 PM"

This allows us without unnecessary back and forth confirmations to make sure that Alexa understood us right, since if she got a wrong airport code for **Cairns (CNS)** instead of **Cairo (CAI)** her answer would be:

"LH 779 arrives to Cairns via Singapore today at 5:40 AM "

It avoids a scenario, where Alexa would sound cumbersome by asking

"Did you just ask about flights from Cairo?"

or even worse

" Are you sure you want to check flights from Cairo today?"

If we want to add session retention to this scenario to make it more relevant, Alexa could also check next time we open the skill if we want to book the flight we just checked since it is likely to be still in our interest.

As for repetition, it makes more sense to remove repeating words that the user has to say and the sentences Alexa says. This can be done by changing boilerplate strings in the code to arrays where it becomes less predictable what Alexa would say next time we open the skill.
Here is a code example from our implementation:

```
GREETING_TEXT: [
'Ich kann dir mit den zahlreichen Dienstleistungen der Stadt Berlin
↪  helfen! ',
'Möchtest Du dich über Öffnungszeiten oder eine Dienstleistung
↪  informieren?',
'Willkommen in dem Hauptstadtportal. Was kann ich für dich tun?'
]
```

```
//choose one Utterance for Alexa to respond with.
exports.getRandomResponseUtterance = function(inputArray) {
const randomUtterance =  Math.floor(Math.random() *
↪  inputArray.length);
return inputArray[randomUtterance];
}
```

To make it even more situation-dependent, think of a Skill for instance that would add to its greeting on friday a

“ Have a nice weekend.”

or if the user was not on the Skill lately, Alexa can present what the Skill can do since his/her last visit.

In short, repetition can make a VUI sound dull or interesting and the choice of what to confirm vs. what to infer is best validated with A-B tests to different user groups.

The Opposite of GUI

Whenever we think of wanting to teach the user to stick to something, we should try to eliminate the thought. With a GUI, it is important to keep the interface consistent. A prevalent example is when Microsoft changed its Office Layout and it was hard to re-teach the users to the new layout, icons and ribbons view. In GUI it is usually preferred to avoid this consistence since it breaks the monotony and the repetition problem mentioned above.

Speech Synthesis Markup Language (SSML) and Other Effects

SSML is a great industry-standard tool to make sound variations and add human-like effects to the conversation and are encouraged. Although not all tags are supported by all systems, Alexa offers an broad range of sounds.

We should hence not exclude use of audio clips etc, music, music and speech together. Alexa also integrates Speechcons. These are certain words or phrases pronounced by Alexa more expressively. Here is an example of how these look different to a string:

```
<speak>
Sometimes when I think of all those best practices, I just say,
<say-as interpret-as="interjection"> Horray. </say-as>
</speak>
```

Speechcons are available in German [2] and English [3]
Also, Polly can change the sound of the character behind Alexa, which is commonly used in games with multiple users.
Timeline markers are a helpful feature with lists as well, e.g. when saying

[2] a͡sk/speechcon-reference-interjections-english-us.html
[3] a͡sk/speechcon-reference-interjections-german.html

“ First, ...”, “Second”, “At the end”

Flavours, Pointers, Acknowledgement of Feedback

Giving the user a feeling that they are not talking to a rigid machine, or even adding more ‘manners’ to the language, like when we want to ask Alexa to buy some clothes for us, she would sound better if she says

“ Sure, what size?”

rather than just

“ What size?”

Adding pointers like ‘this’, ‘that’, ‘your’, ... personalise the experience and wrap up the sentence more elegantly. Additionally, adding transitions like

*“ **Now**, we’re going to talk about ...”*

Building flexible Paths

Also known as the graph- vs. frame-based design problem, since we have no idea what the user will say and cannot limit their choices through certain buttons on a screen etc, we want to be as captive as we can to what they say. For instance, a good Alexa Skill should be able to handle when the user says

“I want {blue} {Nike} {Hi-Tops} ”

and also when they say

“What are the trendy sneakers in stores”

Also considering that the user could get too familiar with the Skill and use even more complex thinking with it, they might in the middle want to restart the search process for the perfect product. The Skill should therefore be flexible enough to cope with that. This translates into a more horizontal design as opposed to just following a workflow

where the user will be asked only about a colour then a size then an item before they get to their product. Used in many advanced interactive voice response (IVR), there is also extensive documentation on this design field that goes beyond the scope of this thesis.

Voice-First, Voice-First, Voice-First, Multi-Modal

Although there is enough emphasis to the voice-first approach throughout this work, it is also important to take advantage of using a screen whenever **necessary**, not whenever **possible**. Making a screen a supplement that the user does not have to depend on is vital to the user experience, since we need to go from the worst case of a lazy or even almost paralysed user who would not immediately jump to a TV screen when they are asked to do so by Alexa. The cards presented on screen should only reinforce what we say and not diverge at any time or provide important information that is not present in voice. This is as much GUI as we want to integrate in our VUI using Alexa and might change in a different context for niched markets with special cases for Alexa or other voice assistants.

Modularizing the Skills

It is sometimes better to build small Skills each performing a single action and connect them together with the same APIs rather than building one gigantic skill that expects too many utterances that could render it futile. One example is to make a Pizza menu Skill and another Pizza order Skill that depends on the data from the Pizza menu app. Each would have a separate `LaunchIntent` enforcing a separation of concerns.

Invocation Name

While Alexa's presence in Germany is fairly recent giving a freer choice of names, it is pivotal to the Skill that it has a name users can remember easily. The key is not just in selecting the key, but also testing what Alexa interprets it for. A Skill name like 'four miles' can be mistranscribed as 'for my's' or 'form isles'. Checking the speech history in the Alexa App (for the end-user) helps determine if these mistranscriptions happen too often.

Designing for Natural Conversation

By having other users test the Skill conversation throughout the development process, many unnecessary scenarios can be circumvented and new ones unveil. The way we speak are blueprints of our verbal biases, as the way we use certain vocabulary is related to where we live, our own ideology or character, and who were interact with. We are used to only our own language or way conversation. As flexible as this can

be, learning from others' missed intents can be helpful at all times. For instance, if we design a Skill to say a first name, someone might make the Skill say 'Günther' or 'Jacks' while someone else could make it say 'Mama' or 'Daddy', which are not in the predefined slot list `AMAZON_DE.FirstName` (we discuss lists thoroughly in Chapter 9). Checking when and how a Skill fails is of great value to enhancing it.

The aforementioned practices are even more powerful once combined. Before accepting or rejecting the trajectory of a guideline's results, we assess its relevance to our actual design.

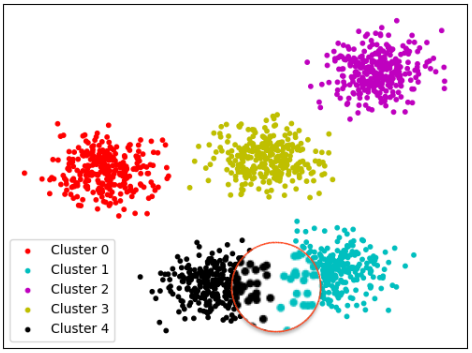
8.2 For GUI Chatbots

While chatbots are not the primary focus of this work, here are a few notable considerations in addition to the ones above that go for Alexa Skills and any chatbot (as long as they do not contradict with the GUI paradigm).

Classification of Intents

Filling intents with expressions is equivalent to building a sentence classifier: Whenever a user sends a new sentence to [our] bot, it will be classified into one of the intents. [We] can visualize [our] bot training as a set of [k-means] clusters: large circles in a two-dimensional space, one circle per intent and incoming sentences are assigned to one of the circles as in Figure 8.1 [24].
The key is making the utterances form non-overlapping clusters by maximising inter-cluster distance and avoid shapes like the overlapping between cluster 1 and 4

Figure 8.1: K-means Clusters of Utterances, based on Deschamps [24]



Keeping it Simple

A Skill or a chatbot that uses 1000 intents or does too many things is more likely to break. If we can break down the tasks into more modular intents / actions, we avoid this risk. Same goes for utterances. In our Skill this could mean we can make one Skill that only books appointments at public authorities and another that only provides information on the services. While it is possible to make a lot of these, it is most of the time better practice to make use of dialogue management and entity resolution and reduces false negatives.

Allowing Surprises

Even when the user says words that do not match to an intent, although we do not want to handle that, we should still be able to respond with an answer that is equally random to the user as the request is to Alexa / the chatbot. For example if we launch a Skill checking for flights and we ask it to order pizza, we can still educate the user about a new thing they were not expected to hear or tell them a joke. Since the user was expecting to challenge or break the Skill/Bo they would get surprised if they end up learning about something new or have a laugh instead.

Moreover, if we are able to find in a log (works in case of using Alexa with Lambda) what are the common 'unusual' uses to our Skill and route them to a response that would make the user laugh or just be surpirsed. Even Better would be if the process works through ML.

Showing Clear Structure

One of the revolutionary changes we witnessed since the introduction of GUIs in any major OS was the 'main menu'. Appearing in the form of the Start Menu on Windows, the Apple Menu and the Dock on the Mac, the functionality of this button was so intuitive, that most keyboards include it as a physical button. Similarly, already with the launch of the first mobile phones, the option of going back to a main screen was always present (with the hangup button) and moved on to more advanced operating systems from most Symbian and Blackberry devices to now the Home Button on smartphones. As it is "second nature for us to look for a home button whenever we want a fresh start" [14]. While using keywords like 'help' or 'start over' can be effective, it depends on the how we train the user to adapt to the use of these words. An effective way of giving the user an idea about a possible next step is to provide answers to the questions the bot gives including a small percentage (i.e. one every max. five answers) that would lead the user to a fresh start or even dedicating an extra button for that function.

In Summary, chatbots and voice assistants are a specific type of computer programmes, which are worth developing guidelines for and following certain design patterns to obtain adequate results. Guidelines differ in a case-by-case scenario based on implementation. Hence, it is important to consider these and include them in the design process.

Chapter 9

Skill Implementation

In this chapter we present our implementation and hands-on experience throughout. The code is available on Github (see Chapter 1) with plans for development beyond this thesis. A copy is attached to this work, but cannot fully operate as a standalone software. The runtime environment is offered through the Alexa Skills kit There are many ways to perform setup, several languages we can use to programme our Skill and throughout the development process some of these steps were changed. We present the most recent interfaces and implementation possibility using Node.js hosted on AWS Lambda

9.1 Setup

Minimum Requirements

For the environment setup, the following tools and credentials are necessary:

Amazon Account

We use it in conjunction with AWS and the Alexa Developer Console. Any account previously created for the Amazon store is usable [1] as well as to test our skill. Once the account is activated on the developer console, the Skills in development become automatically reachable on any Alexa-enabled device linked to the same account (as an end-user).

Command Line Tools

- **Python 3** or **Ruby 4** need to be installed as a foundation for ASK CLI and AWS CLI.

[1] e.g. from here `https://aws.amazon.com/de/resources/create-account/`

- A package manager such as Pip Installs Packages (**PIP**) for Python or **RubyGems** for Ruby.

- Depending on OS, further packages and package managers might need to be installed. (e.g. for macOS: Command-Line Tools for Xcode, Homebrew, easyinstall). These will be prompted throughout the installation process.

API

As introduced in Section 7.1, Our endpoint is a Solr Server that provides us with JSON responses to the queries we do through a RESTful interface from within the Lambda function (hosting back-end code) [2]. Since the API endpoint was originally created for the Virtual Citizen Assistant, we need to readjust its output to meet Alexa's requirements.

Audio

An encoder to create 48kpbs constant bitrate MP3-Format. Audacity [3] or ffmpeg[4] can be used to convert to an Alexa-friendly file format like in the ASK DOCUMENTATION[5].

Managing AWS Credentials

With the Amazon account we just created (or already have), we will proceed to extend it for developer registration. As we move within the free tier of AWS, no charges were applied. However, a credit card might be required for eventual future billings if the tier limitations were exceeded. We mention pricing schemes in Section 5.3 [6]. Developer registration needs to happen on AWS website[7] as well as on the Amazon Developer Portal[8] where a some information is required such as company name and business sector. From this point on, anything that manages the back-end of our skill will be associated with AWS Modules, primarily Lambda. Skill configuration and interaction model are hosted on Alexa Skills Kit separately, i.e. managed through the Amazon Developer Console.

[2] reachable at the time of development on:`https://newsreel-edu.aot.tu-berlin.de/solr`

[3] `http://www.audacityteam.org/`

[4] convert by running `$ ffmpeg -i <inFile> -ac 2 -codec:a libmp3lame -b:a 48k -ar 16000 <outFile.mp3>`

[5] `https://developer.amazon.com/de/docs/custom-skills/speech-synthesis-markup-language-ssml-reference.html#h3_converting_mp3`

[6] More information on pricing available on: `https://aws.amazon.com/de/pricing/`

[7] `http://aws.amazon.com`

[8] `https://developer.amazon.com`

During the development process, Alexa Skills Kit diverged from the Amazon Developer Console and has become a separate entity called Alexa Console. While still hierarchically belonging to the Amazon Developer Console and reachable from the same domain, except for the developer profile, all skill-related information we need to use are modifiable from within the Alexa Console[9]

Generating Access Keys

Linking happens through a private and public key pair - the **access keys** consisting of a combination of an *access key ID* (like `AK..7EXAMPLE`) and a *secret access key* (e.g `wJalrXUt..I/K7..G/bPx..YEXAMPLEKEY`). We use access keys to sign API requests that you make to AWS.

Creating Access Keys is described in the AWS DOCUMENTATION[10]. At this time we interact with the IAM Module from AWS (Identity Access Management), which defines roles of each member in an organisation and partners linked to it, who can gain different levels of access to each module capabilites. Since we are only making basic use of AWS modules and our Lambda instance interacts only with our own API, there is no harm in using

Initialising The Development Environment

As minimum package requirements can vary a lot between one OS and another, using a virtual machine is suggested for teams. AWS offers Elastic Compute Cloud (EC2) as a service for virtual machines in the cloud. Although this step is mainly for performing command-line prompts later on, if opting for that, it is convenient to avail a web browser inside the virtual machine instance. Otherwise for browser communication the tool we are going to use will generate links that we would need to copy and paste into a browser outside of the virtual machine to link it with the Amazon account we use.

Installing Command-Line Tools

```
1   $ which node      #check if installed
2   $ which npm       #check if installed
3   $ which python    #check if installed
4   $ which ruby      #check if installed if proceeding with rbenv
5   $ python --version #we use 3.5.3
6   $ node --version   #we use 7.3.0
7   $ npm --version    #we use 4.1.1
```

[9] https://developer.amazon.com/alexa-skills-kit

[10] https://docs.aws.amazon.com/general/latest/gr/managing-aws-access-keys.html

```
8   $ pip install aws-cli #Amazon Web Services Command Line Interface
9   $ aws --version    #check successful installation. We use
    ↪  aws-cli/1.15.1 Python/3.5.2 Darwin/15.6.0 botocore/1.10.1
10  $ npm install ask-cli| #Alexa Skills Cit Command Line Interface
11  $ ask --version    #we use 1.1.6
```

Understanding the API Query parameters

HTTP(S) endpoint: `http://newsreel-edu.aot.tu-berlin.de/solr/#/d115`

Rest Query Solr Syntax (only url. he knows solr better than i do)

step by step code analysis

`https://github.com/alexa/alexa-skills-kit-sdk-for-nodejs/blob/master/Readme.md`

for audio (old)

json format https://stackoverflow.com/questions/41776014/how-to-correctly-specify-ssml-in-an-alexa-skill-lambda-function

references:

`https://developer.amazon.com/docs/custom-skills/speech-synthesis-markup-language-ssml-reference.html#audio`

blogs: `https://developer.amazon.com/post/Tx3FXYSTHS579WO/Announcing-New-Alexa-Skills-Kit-ASK-Features-SSML-Audio-Tags-and-Developer-Porta`

With the JSON we have, we try to maximise its use, but we need to consider that with starting a Skill, it is better to start from scratch sometimes and as we go, we see what is the relevant information that we can integrate into our Skill.

We analyze the output of the VIRTUAL CITIZEN ASSISTANT and realize it makes less sense to start with that, so we refer to the underlying endpoint. Since this is the Solr Server that delivers JSON objects and is more realistic to maneuver, we analyse the queries we can get from there. and try to fit it into our interaction model.

Then we move on to the interaction model. On paper, we draft the use-cases to determine what are our intents first to group these into fewer intents than the services we have. Given that there are many public services related to a similar case, we design our Skill to make it possible to group them into less intents. The advantage of this is that it allows us to at least have fewer 'ServiceIntents' than the ever growing number of services, which means that we want to also be able to track new services as they get inserted into the catalogue to be able to either map them to old intents by extending these, or to introduce new intents if this does not work.

✍ **TO-DO**: moved from intro

our now more than ⌈ 4 ⌋ scenarios:

- a general scenario of predefined categories related to any service **[HIER SCHON ERWÄHNEN?** ⌈ (prerequisites, costs, documents, ...) ⌋ **]**
- special application on **Residence Registration** - "Anmeldung einer Wohnung"
- special application on **Applying for a Residence Permit** - mutliple services related to "Aufenthaltstitel"
- "car registration" - special application on **Car Registration"** - multiple services related to "Kraftfahrzeug (KFZ)"

✍ **TO-DO**: moved from AWS Micro-services list

We might as well use just our own, any other privately or cloud-based server solution to host our code and use it as an endpoint for our software, however choosing Lambda takes away the overhead of linking the front-end with the back-end of the Skill

start with interaction model then use builder then fulfill functions

steps to git commit deploy clone change in browser change offline

if u use a browser, remember to keep the session active otherwise your model might get lost in the POST

✍ **TO-DO**: this goes after we have a lambda instance connected to our skill

Verify through Application ID [11]

9.2 Code Analysis

from `https://github.com/alexa/alexa-skills-kit-sdk-for-nodejs/blob/master/Readme.md` The difference between :ask/listen and :tell/speak is that after a :tell/speak action, the session is ended without waiting for the user to provide more input. We will compare the two ways using response or using responseBuilder to create the response object in next section.

A LaunchRequest always starts a new session.

jsons sent/received in appendix 11.1

[11] ask`/handle-requests-sent-by-alexa.html#request-verify`

legal things like the EU-partner rule. eu rules override local bas u cant explain it to people 3ashan they have to be give the choice. el mawdou3 mo3aqqad but codified here https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:02004L0038-20110616

9.3 Feature Implementations

Creating SSML and Audio

ask/speech-synthesis-markup-language-ssml-reference.html#audio

Checking the Next available appointment

Although this is a proof of concept, with an API extension from ITDZ's side, it is equally easy to implement a feature to look up next available appointments. The actual booking would require Alexa to send POST requests and persistently retain these on the API's side with the option to impement another feature to save the same appointment to the calendar linked to the Amazon account used with this Alexa instance. This might require account linking, too, if we need to pass the appointment data through the Skill to an online personal calendar endpoint, bypassing Alexa itself.

Booking an appointment happens for now only through Berlin.de's website by starting at either the appointment booking page [12] which links to the actual booking pages via choosing the service or the location where the service

Card Templates

Rendering the right template or a custome one [13]

Utterances generator

http://alexa-utter-gen.paperplane.io/ https://github.com/tejashah88

✍ **TO-DO**: u

nhandled always calls or never

[12] https://service.berlin.de/terminvereinbarung/termin/
[13] display-interface-reference.html

9.4 Testing

continuous testing during development, noch nicht evaulation, although the same techniques are pretty much used afterwards. Requires 10 users a day to generate

✍ **To-do**: history API - Dave isbitzkis tweet

```
$ ask api intent-requests-history <-s|--skill-id <skillId>> [--filte
```

https://developer.amazon.com/docs/smapi/ask-cli-command-reference.html#intent-requests-history-subcommand

https://developer.amazon.com/docs/smapi/intent-request-history.html

9.5 Deployment

9.6 Approval and Publishing

Continuous Skill Approval is the process of Amazon deciding whether each uploaded version / update is still fit for purpose, make the skill better or worse in conjunction with other skills on the Store

✍ **To-do**: mention that 3ashan homa store w keda fal mawdou3 lazem yekoun mo7kam w 7agat homa bey7ebouha and the reasons why things could get rejected according to their policy and some scenarios

9.7 Debugging / Troubleshooting

✍ **To-do**: API TLS certificate. No base64 just `rejectUnauthorized: false`

9.8 Documentation Updates

https://www.gitbook.com/join/tu-berlin/-LAZYgGmoeaEZGb8cYg2 https://tu-berlin.gitbook.io/alexa/

- as an example for text
- implementing the answer suggestions as buttons
- passing data to the Bürgeramt terminseite

https://console.dialogflow.com/api-client/

https://console.actions.google.com

Solr has to be secure https://docs.aws.amazon.com/AmazonS3/latest/dev/WebsiteEndpoints.html#WebsiteRestEndpointDiff

Hey:

do this set up as prerequisites first as a deployment script

- install aws cli - install ask cli - set up a profile on AWS - ask init etc.

https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-getting-started.html

There is no AWS credential setup yet, do you want to continue the initialization?

https://developer.amazon.com/docs/smapi/quick-start-alexa-skills-kit-command-line-interface.html

Manage IAM roles (User, groups, rights, policies etc.)

all logs go onto cloudwatch including when the request is not right etc...can be helpful to know what users want, beyond testing

amazon does not disclose avaliable values for their slot types (lists), but they give you examples here

===

finally, we had to resolve to the information publicly available to tailor custom scenarios

=== once you upload the new lambda, the old one is gone, unless you version it like this: https://docs.aws.amazon.com/lambda/latest/dg/versioning-aliases.html

====

first i got screwed over with this Big nerd ranch then this happened: alexa-skill module - so don't use this tutorial, although very helpful for a beginner and makes you understand what happens under the hood, it has been abstracted in many other function and the logic is no longer the same. - which explains why no one starred it

https://github.com/matt-kruse/alexa-app

https://www.bignerdranch.com/blog/developing-alexa-skills-locally-with-nodejs-deploying-your-skill-to-staging/

✍ **To-do**: OnLaunch
IntentHandler
intent is triggered by utterence
account verlinkungen etc

Prob with outdated tutorials

✍ **To-do**: eventually mention here:
as we mentioned Amazon's strong take on constant changes, we will go over the workarounds and circumventions to retain a working version of the Skill we present, discuss the downside of adaptability to these changes ⌈ (constantly basastem nafsi 3ala 7aga gedida) ⌋

Chapter 10

Evaluation

10.1 Comparison with Existing Systems

WienBot

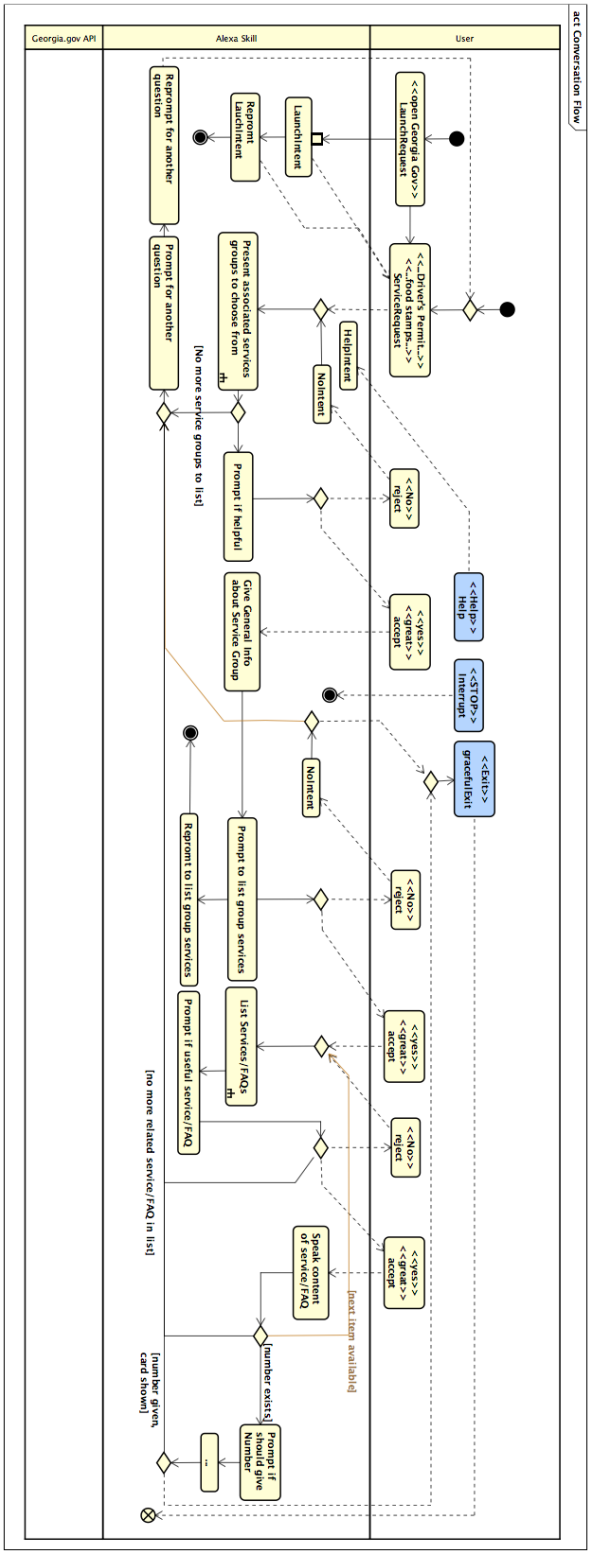
✍ **TO-DO:** WienBot - der Chatbot der Stadt Wien.pdf
-versteht nicht, wenn ich 'perso' schreibe
-verbose.. "warum möchten Sie aus der Stadt...
-uses emojis
-speech recognition happens on device (iOS)
-use failed attempts of screenshots (new and old)

GeorgiaGov

- Talk about features like Maintaining context(sesions), **Intent chaining**

child care vs child custody. Sorry i didnt get that. (check first skizze)

Figure 10.1: Activity Diagram of Conversation Flow on Georgia.gov



✍ **TO-DO**: -This is how their model works:
services are grouped into categories as seen on georgia.gov
-Alexa catches from any sentence you say only the service grouping name.
-then gives you a definition
-then asks you if you want to hear about related services (in the format of FAQ)
-possible examples: “I am 17 and my parents are blind. How do I renew my license?”
-it goes through this list and you can say yes and no if you want to hear about the ‘related’ item.
-of the related items are questions like ‘how much does it cost’, ‘how much time does it take’. These are no explicitly put in the questions you can ask but they become clear if you have time to make conversation. (Kritikpunkt!) -then when it reaches the end of the list it says that it’s over and if you want to have the phone number that is related tot this service grouping -then the conversastion either ends or restarts.
-you can ask questions only about service groupings
-stop and cancel intent: *Feel free to ask another question or say exit* -helpIntent: Ask me a question repeat does -reprompts are not very ueful and repetitive

This way, if a user does not see this service is relevant to them, they will say no and skip to the next

10.2 Alexa in General

Experian of Alexa as a whole `https://www.experian.com/innovation/thought-leadership/amazon-echo-consumer-survey.jsp`

✍ **TO-DO**: are Skills really going to make it?
30,000 users a day in the US
`https://omr.com/en/amazon-alexa-skill-marketing/` 10,000 BVG

✍ **TO-DO**: metrics from Seif’s BA!

✍ **TO-DO**: sometimes voice will hardly or never get what you want. try to tell alexa to play YAS. will always redirect you to something else

✍ **TO-DO**: - wo hilft mir alexa, was mach ich lieber woanders?
- wie kann man die Güte des Systems beurteilen? **do not forget the surveyu made**

10.3 Metrics

✍ **TO-DO**: **talk about umfrage-design:**
skalen, yes no, choosing between limited answers, etc.
combining logic and validation, such that you would be asked about only one thing if you answered the other respectively. done in english and german for purpse: cater for different people living in berlin
conjoint analyse
a-b testing - see what users prefer then come to conclusion that dienstleistung is the most important
we evaluate how people speak to alexa with sample sentences.
ask questions like "Did u know berlin.de has a behindertenberechtigt" to educate the person at the same time

✍ **TO-DO**: what disturbs you most about alexa?
would u try it again if the question fails?
-thats why the retention rate is low

✍ **TO-DO**: NPR
time series a-b testing weighting use of correlations in the legal system (got, defaulting) search map predicts ur personality fear of judging, fear,. . . interdependence with robots likelihood of having different social cirlces for a lasting relationship
- see how many use the skill after publishing tinyurl.com/AlexaBln
`https://umfrage.hu-berlin.de/index.php/879458?lang=en`
`https://umfrage.hu-berlin.de/index.php/879458?lang=de`

-benchmarks
-strengths and weaknesses
-challenges
-performance
-usability
-feasibility of using the studied agents
- node.js?
- amazon's system testing options (incl. Betas)
-
- system usability scales (ISO, DIN)
- Con: Alexa skills are listed in the amazon shop page. Sehr unübersichtlich
just like prime
- impression: Amazon collects data and makes something "intuitive out of it for you". e.g. fire stick setup already had account linked before connecting to the internet!

scary/funny/ but then it could be counterintuitive at some point if u want to do ur own customizations.
- removing bias in recriutment of participants (diversify based on what categories?)
-
- EVAL: AUC/ROC, true positives, false...no of utterances to text
- compare with Wiener Stadportal as a benchmark for a bot
https://www.wien.gv.at/bot/ http://www.vienna.at/wienbot-chatbot-der-stadt-wien-informiert-als-virtueller-beamter/5590853 https://digitalcity.wien/wienbot-auszeichnung-fuer-chatbot-der-stadt-wien/ singaporebot

10.4 Results

mention git commits?

usability metrics: - heuristic eval - guidelines **(jakob nielsen, ralf molich whitepaper)**
– biggest usability flaw
- cognitive walkthrough
– step-by-step approach
– questions..wil the user tr and achive
- pluralistic walkthrough
– panel method
- hallway testing
- A/B Test
- speed and Bottlnecks
-
- clientele: census / SOEP, who can use the bot
- make a small prediction (Bus Analytics)
- this Hassloch thing from MKTG

10.5 Discussions

✍ **To-do**: will never get things like A$ap Rocky, Y.A.S elli 3ala spotify not ay wa7da, or Ta3ala (egyptian arabic)

✍ **To-do**: you mentioned in intro (structure of this thesis) that you will discuss ML approaches like term weighting

✍ **TO-DO**: wienbot screenshots
alexa app caught what i said screenshots

alexa ask georgia if u cant install it, here is a video https://vimeo.com/216737044

while it may seem trivial at the start to set up, in den details steckt der teufel. from installing npm to ... and the flexibility of the system - so many options to choose from: swagger, dynamo,...

- Evaluate the system:
- is it trivial to build such a bot or not / what is the aufwand
- how does it react with longer sentences? some service names are long
- what does levenstein distanz cause
- wie leicht kann ich eine antwort finden auf das was ich suche?
- how am i going to classify my tests?
-
- are chatbots being pushed on the market or is there a demand? (kleine Umfrage basteln?)
- how easy or difficult it is to make a bot: planing poker - varianz anschauen zw. leicht und schwer und iterativ darüber sprechen
- wo kann der Kunde (Sawa2 kan el end user or the senat in our case) help optimize the bot masalan bürgeramt beyektebo, welche Rechtsgrundlage k̈eine-̈ auffällige Probleme masalan zay Perso, PA, personalausweis, how to introduce ëxpert modeš̈o that if u add it with a special character it knows what u want, just like alexa knows when u rename the lamp - refer again to use cases and exper vs personal field

✍ **TO-DO**: mention that building with alexa allows entertainment through small talk and assures the integration of the skill in a widely used platform, but small talk gets you out of the skill in the pro/con section. it's not a con but it needs to be addressed. like meaning that u develop an app for iPhone is not enough guarantee that having an iPhone makes people only use ur app.

Chapter 11

Conclusion and Future Work

11.1 Overview

Chatbots and voice assistants are raising debates about sexism in AI, their ability to supersede human thinking, our acclimatisation to their presence. They symbolise chances and threats and are unlikely to be indifferent to. All of which show signs of a disruptive industry that brings a major change forward. And while their use to replace humans in different jobs is becoming reality, it can be regarded as a change that brings other changes with it since it also opens up unprecedented job opportunities in technical and non-technical fields.

✍ **TO-DO**:
at alexa somewhere **i.e. talk about SKILLS**
- ability to retain sessions (explain requests/responses - GET/POST)
- fullfilling intents
- nested handlers

skill service: code - business logic - handles json requests skil interface: configuration (developer portal)
- difference to Lex & Polly : diff alexa lex

- Major prob: lex is not in german
- Alexa Documentation

suitable
mention the D115 how u met them
IMPORTANT table with fara3in (Nodes):

✍ **TO-DO**: non-fluid text, as in not human readable. we see what it means later with the SSML, add string in line

kurze sätze to Alexa 3ashan mateghlatsh

http://www-personal.umich.edu/~cellis/heteronym.html homophones, homographs moved from ch2 Underlying Grammar: since languages do not work with a unified grammar and word types (adjectives, adverbs) are not simply replaceable by one another. If a person say I a

from ch 2: And so, naturally, what distinguishes a good system from a better one is how fast it learns.

✍ **To-do**: That's why it's good that there is more than one giant taking care of this

11.2 Conclusion

✍ **To-do**:
-matensash el evaluation beta3et el chatbot

- kritikk that amazon is v haphazard. zB memo in workshop

Retention Dos and Don'ts

✍ **To-do**: beyond the features, we can take something from wienbot: they are active about marketing their bot
mention what aspects of the model we can mock. e.g. their wienbot twitter feed re new launches. https://futurezone.at/digital-life/wienbot-die-stadt-wien-hat-einen-chatbot/246.709.043

11.3 Future Work

- use machine learning to rank higher demands for more popular services.
- matkhoshesh fel 7etta di awi - for now hitlist already given.
- future of bots. deren Einsatz. roles (As judges, catereres in hotels (that hotel botler)
-

✍ **To-do**: We are going to build neurons so they can bridge together (Alexa Podcast 18) Otherwise they will forget Branches stop growing when they reach another branch Intelligenceaggreation - amplification We are voice-first approach (as said earlier: Inner voice) We do not know much about human brain /intelligence- -the point is not in taking over but making Internet more powerful (Podcast) information will appear as you need it: Voice first not voice only - less time to look at screen.

This derives into a possible scenario where Alexa's AI becomes smarter then maybe in a few years the concept of Skills won't exist as a stand-alone any more and would be integrated into Alexa's own brain without the user knowing and it would just be a hit or miss kind of thing.

Bibliography

[1] Adam Marchick, Alpine.ai. Voice search trends. https://alpine.ai/voice-search-trends/, April 2018. Online Resource. Accessed on 16.04.2018.

[2] Alpine.ai (VoiceLabs). The 2017 voice report. https://alpine.ai/2017/01/15/the-2017-voice-report-by-alpine/, January 2017. Online Resource. Accessed on 16.02.2018.

[3] Amt für Statistik Berlin-Brandenburg. Statistischer Bericht - A I 5 hj 1. https://www.statistik-berlin-brandenburg.de/publikationen/stat_berichte/2017/SB_A01-05-00_2017h01_BE.pdf, August 2017. Online Resource. Accessed on 06.02.2018.

[4] George Anders. "Alexa, Understand Me". https://www.technologyreview.com/s/608571/alexa-understand-me/, August 2017. Online Resource. Accessed on 21.04.2018.

[5] Annett Witte, Friedrich-Naumann-Stiftung für die Freiheit. Freiheit.org - Digitalisierung. https://http://shop.freiheit.org/download/P2@591/97725/FreiheitDigital_02_web.pdf, June 2016. Online Resource. Accessed on 28.04.2018.

[6] Darren Austin. How amazon's alexa hooks you. https://venturebeat.com/2017/06/27/how-amazons-alexa-hooks-you/, June 2017. Online Resource. Accessed on 30.04.2018.

[7] Brian Roemmele, Dave Isbitski. Episode 018 - A Voice First Future with Brian Roemmele. *Alexa Dev Chat Podcast*, November 2017. Online Resource. Accessed on 13.02.2018.

[8] Brian Roemmele, Dave Isbitski. Episode 019 - Looking back at the Year in Voice and What's Next. *Alexa Dev Chat Podcast*, November 2017. Online Resource. Accessed on 13.02.2018.

[9] Wikipedia contributors. History of natural language processing — wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=History_of_natural_language_processing&oldid=795750930, 2017. Online Resource. Accessed on 14.02.2018.

[10] Wikipedia contributors. Model-view-controller. https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller, March 2017. Online Resource. Accessed on 12.02.2018.

[11] Wikipedia contributors. Usage share of operating systems — wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Usage_share_of_operating_systems&oldid=825514016, 2018. Online Resource. Accessed on 14.02.2018.

[12] Alexa Skills Kit documentation. https://alexa.design/build.

[13] Dries Buytaert. https://dri.es/think-beyond-with-acquia-labs.

[14] Eunji "Jinny" Seo. 19 Best UX Practices for Building Chatbots. https://chatbotsmagazine.com/19-best-practices-for-building-chatbots-3c46274501b2, January 2017. Online Resource. Accessed on 10.03.2018.

[15] Gartner. Gartner reveals top predictions for it organizations and users in 2017 and beyond. https://www.gartner.com/newsroom/id/3482117, October 2017. Online Resource. Accessed on 15.02.2018.

[16] Georg Berger, RBB Online. Geniale köpfe - wohnen der zukunft. https://rbb-online.de/abenschau/archiv/20180416_1930/sahin-Albayrak.html, April 2018. Online Resource. Accessed on 16.04.2018.

[17] Geschäfts- und Koordinierungsstelle 115 im Bundesministerium des Innern. https://www.115.de/SharedDocs/Publikationen/Service_Publikationen/Infomaterialien/infobroschuere_%20BMI08324_screen_barrierefrei.pdf?__blob=publicationFile&v=8, November 2013.

[18] Tobias Grossmann, Regine Oberecker, Stefan Koch, and Angela D Friederici. The developmental origins of voice processing in the human brain. *Elsevier*, 65:852–8, March 2010.

[19] Hira Niranjan, Amazon Web Services Webinar Channel. Introducing amazon lex: Service for building voice/text chatbots - march 2017 aws online tech talks. https://youtu.be/tAKbXEsZ4Iw?t=4m14s, March 2017. Online Resource. Accessed on 15.04.2018.

[20] Privacy International. State of privacy egypt. Online Resource. Accessed on 18.01.2018.

[21] Jan Grundmann, SearchMetrics. Voice search im seo-check: Liest google home immer das erste suchergebnis vor? https://blog.searchmetrics.com/de/2017/08/09/google-home-analyse-voice-search/, August 2017. Online Resource. Accessed on 30.04.2018.

[22] Judy Goldsmith and Nicholas Mattei. Science fiction as an introduction to ai research. https://pdfs.semanticscholar.org/fdd1/41aac65e9c412e7804b930ca81db327c26b5.pdf, 2011.

[23] Will Knight. Ai's language problem. https://www.technologyreview.com/s/602094/ais-language-problem/, August 2016. Online Resource. Accessed on 26.04.2018.

[24] Lucas Deschamps. The Art of Bot Design. https://recast.ai/blog/art-of-bot-design/, November 2017. Online Resource. Accessed on 10.03.2018.

[25] M. Ballve. Messaging apps are now bigger than social networks. http://www.businessinsider.de/the-messaging-app-report-2015-11?r=US&IR=T, September 2016. Online Resource. Accessed on 17.02.2018.

[26] Malte Schwarzer, Jonas Düver, Danuta Ploch and Andreas Lommatzsch. An Interactive e-Government Question Answering System. In *Proc. of the LWDA conference, Track Information Retrieval, Potsdam, Germany*, LWDA '16, pages 74–82. CEUR Workshop Proceedings, 2016. Online Resource. Accessed on 09.05.2018.

[27] Matt Anthes, Forbes Agency Council. Social Media As A Vital Engagement Platform For Government Outreach. https://www.forbes.com/sites/forbesagencycouncil/2017/10/02/social-media-as-a-vital-engagement-platform-for-government-outreach/#72dcbe864b29, October 2017. Online Resource. Accessed on 01.05.2018.

[28] Meike Zehlike, Francesco Bonchi, Carlos Castillo, Sara Hajian, Mohamed Megahed, Ricardo Baeza-Yates. FA*IR: A Fair Top-k Ranking Algorithm. https:

//arxiv.org/abs/1706.06368, January 2017. Online Resource. Accessed on 29.04.2018.

[29] Memo Döring. Alexa dev days. http://alexadevday.com/BER/DevPortal, April 2018. Remarks in Workshop.

[30] MindMeld, Cisco Systems. Intelligent voice assistants research report, q1 2016. http://info.mindmeld.com/survey2016q1.html, 2016. Online Resource. Accessed on 30.04.2018.

[31] Mohamed Megahed and Vasilis Ververis. Surveillance, Blocks and Phishing in the Nile, March 2018.

[32] Mohammad El-Taher. Closing Windows.. Censorship Of The Internet In Egypt. https://afteegypt.org/digital_freedoms-2/2018/02/19/14655-afteegypt.html?lang=en, February 2018. Online Resource. Accessed on 11.05.2018.

[33] Natasha Lomas. Amazon and Microsoft agree their voice assistants will talk (to each other). https://techcrunch.com/2017/08/30/amazon-and-microsoft-agree-their-voice-assistants-will-talk-to-each-other, April 2018. Online Resource. Accessed on 30.04.2018.

[34] NBC. Amazon's alexa had a breakout holiday — people even used echoes to buy more echoes. https://www.cnbc.com/2017/12/26/how-many-amazon-alexa-echoes-were-sold-over-the-2017-holidays.html, December 2017. Online Resource. Accessed on 15.02.2018.

[35] Ryan MacInnis. Advice on Building Voice Experiences from Alexa's Former Head of Product, December 2017. Online Resource. Accessed on 29.04.2018.

[36] Sebastian Hansmann, Marcel Müller. Chatbots in egovernance, 2017.

[37] Seif Osman. Comparison of intelligent personal assistants, 2016.

[38] Amazon Web Services. https://aws.amazon.com/.

[39] Shankar Vedantam. Radio Replay: I, Robot. https://www.npr.org/podcasts/510308/hidden-brain, January 2018. Online Resource. Accessed on 25.01.2018.

[40] Statista. Smartphone user penetration as percentage of total global population from 2014 to 2020. https://www.statista.com/statistics/203734/global-smartphone-penetration-per-capita-since-2005/, November 2016. Online Resource. Accessed on 06.02.2018.

[41] Statistisches Bundesamt. : E-Government-Angebote der Behörden oder öffentlichen Einrichtungen, January 2016.

[42] Laurie Sullivan. Gartner predicts 30 https://www.mediapost.com/publications/article/291913/gartner-predicts-30-of-searches-without-a-screen.html, December 2017. Online Resource. Accessed on 12.2.2018.

[43] Tim Hearn and Paul Bockelman. Transform government it with vmware cloud on aws, an integrated hybrid solution, April 2018. Quotes from attended conference.

[44] Tomas Laurinavicius, Forbes Tech. 20 Fun Facebook Messenger Bots To Play With Right Now. https://www.forbes.com/sites/forbesagencycouncil/2017/10/02/social-media-as-a-vital-engagement-platform-for-government-outreach/#72dcbe864b29, April 2017. Online Resource. Accessed on 02.12.2017.

[45] Trushar Barot and Eytan Oren. Guide to Chat Apps. https://www.npr.org/podcasts/510308/hidden-brain, November 2015. Online Resource. Accessed on 18.02.2018.

[46] United Nations Department of Economic and Social Affairs. E-Government Survey 2016, E-Government in Support of Sustainable Development. http://workspace.unpan.org/sites/Internet/Documents/UNPAN97453.pdf, February 2016.

[47] Voysis, Retail TouchPoints. Voice grows louder: Retailers prepare for the rise of voice search in mobile commerce. http://voysis.com/reports/voice-grows-louder-report-voysis.pdf, 2017. Online Resource. Accessed on 29.04.2018.

[48] Joseph Weizenbaum. *Computer Power and Human Reason: From Judgment to Calculation*. W. H. Freeman & Co., New York, 1976.

[49] Katie Young. https://blog.globalwebindex.net/chart-of-the-day/25-of-16-24s-use-voice-search-on-mobile/, June 2016. Online Resource. Accessed on 16.02.2018.

Appendix

✍ **TO-DO**: check bibstyle - check all relevant docs in Lit review folder are included

On Etymology of Product Names

Product naming is a branding problem which companies deal with differently. In the case of Apple, with exception of the 'Apple Watch' and on their consumer products line, they opt for a clever version using the letter 'i' in front of simple common words, like iBook, iPad, iPhone. though this naming convention originally arose from 'i' to stand for 'interactive' celebrating the early adoption of the internet on the Macintosh product line, it has turned into a naming breakthrough since it had two effects: 1) it gives the user of Apple software or hardware product a feeling of personalisation for the letter 'i' would make something sound as belonging to oneself, namely " 'I', as a person" as a prefix for the name of the object create the name of the product, and 2) it is a marketing strategy to associate all products and services carrying the prefix 'i' with Apple without the need to name the company's name. This applies to products and services that came to life even long after people's adoption of the internet and the widespread use of Apple products, like with iCloud as a web service. Another example is with professional software products like AutoCAD combining the company's name (AutoDesk) with the product's functionality (Computer Aided Design). Companies like Adobe clearly prefer to name their products and services with their brand's full name included, as the case with Adobe Photoshop, Adobe Flash (formely know as MacroMedia Flash). Similarly, as the youngest of them, Amazon adopts a combined approach by stating the full name of the company on its consumer line like with 'Amazon Prime', 'Amazon Alexa' and a mix between 'AWS' and 'Amazon' like with 'Amazon SageMaker', 'AWS Firewall Manager' etc. on its developer B2B line.

Difference Between Lex and Alexa Skills

Amazon Lex is a service for building conversational interfaces using voice and text. Powered by the same conversational engine as Alexa, Amazon Lex provides high quality speech recognition and language understanding capabilities, enabling addition of sophisticated, natural language ćhatbotsźo new and existing applications. Amazon Lex reduces multi-platform development effort, allowing you to easily publish your speech or text chatbots to mobile devices and multiple chat services, like Facebook Messenger, Slack, Kik, or Twilio SMS. Native interoperability with AWS Lambda, AWS Mobile-Hub and Amazon CloudWatch and easy integration with many other services on the AWS platform including Amazon Cognito, and Amazon DynamoDB makes bot development effortless. [1]

[1] More information: `https://stackoverflow.com/questions/42982159/differences-between-using-lex-and-alexa#URL`
`https://aws.amazon.com/about-aws/whats-new/2017/09/export-your-amazon-lex-chatbot-to-the-alexa-skills-kit/`
`https://aws.amazon.com/lex/faqs/`

Documentation (Base) URLs

Amazon Web Services - aws

AWS Glossary	https://docs.aws.amazon.com/general/latest/gr/glos-chap.html

Alexa Skills Kit - ask

Documentation	https://developer.amazon.com/docs/custom-skills/

GitHub Repositories - GH

A Skill Template with API Calls:	https://github.com/skilltemplates/api-starter-alexa
Node.js code	https://github.com/alexa/alexa-skills-kit-sdk-for-nodejs
Building Responses in Node.js	https://github.com/alexa/alexa-skills-kit-sdk-for-nodejs/wiki/Response-Building
Alexa Cookbook (maintained by Amazon Team)	https://github.com/alexa/alexa-cookbook/
Handling Responses (Multi-turn conversation)	https://github.com/alexa/alexa-cookbook/tree/master/handling-responses
SSML Audio	https://github.com/alexa/alexa-cookbook/tree/0afbfcb8ddcd911e83216b4528b20f3a1628c99f/handling-responses/ssml-audio

Outtakes from a query response read as JSON by the Skill code (Solr Response for Table 3.4)

HTML Tags inside Text

```
{ "d115Description": "Eine Fiktionsbescheinigung wird ausgestellt,
↪  wenn über einen beantragten Aufenthaltstitel noch nicht
↪  entschieden werden kann, z. B. weil<br />\n<br />\n<ul
↪  class=\"list\"><li> ... Ausländerakte ... liegt,</li> ...
```

Ambiguous words that can be a plus for the Virtual Citizen Assistant but become irrelevant or confusing for the Skill (e.g. `Fiktionsbescheinigung`)

```
{... "d115Synonym": [
"Aufenthaltserlaubnis","Aufenthaltstitel","vorläufig",
"Visum","Fiktionsbescheinigung"] ...
```

Inconsistent or unconventional text encoding (e.g. `::: false :::`)

```
{... "d115InfoLaw": ["§ 81 Aufenthaltsgesetz - AufenthG  ::: false
↪  ::: http://www.geset...net.de/aufenthg_2004/__81.html"] ...
```

Text gets lost in translation from reading to speaking or is hard to read out loud

```
{... "d115Prerequisites": [ "Persönliche Vorsprache ist erforderlich
↪  ::: false ::: ",
"Rechtmäßiger Aufenthalt mit oder ohne Aufenthaltstitel ::: Zum
↪  Zeitpunkt des Antrags muss die Antragstellerin oder der
↪  Antragsteller entweder<br />\n<br />\n<ul
↪  class=\"list\"><li>einen gültigen Aufenthaltstitel
↪  (Aufenthaltserlaubnis oder nationales Visum für längerfristige
↪  Aufenthalte - Kategorie D - ) [...]",
"Antrag auf Aufenthaltstitel ::: Eine Fiktionsbescheinigung wird nur
↪  dann ausgestellt, wenn [...] ::: ",
"Hauptwohnsitz in Berlin ::: false ::: "] ...
```

```
{... "d115Requirements": [ "Bisheriger Aufenthaltstitel ::: Soweit
↪  vorhanden, ist der bisherige Aufenthaltstitel mitzubringen, z.B.
↪  der elektronische Aufenthaltstitel (eAT). ::: ",
"Nachweis über den Hauptwohnsitz in Berlin ::: <ul
↪  class=\"list\"><li>Bescheinigung über die Anmeldung der Wohnung
↪  (Meldebestätigung) <strong>oder</strong></li><li>Mietvertrag und
↪  Einzugsbestätigung des Vermieters</li></ul>Mehr zum Thema im
↪  Abschnitt „Weiterführende Informationen\ ::: ",
"..."] ...
```

Not a single readable item price


```
{...
"d115Fees": "Ab dem 01.09.2017:<br />\n<br />\n<ul
↪  class=\"list\"><li>Für Erwachsene: 13,00 Euro</li><li>Für
↪  Minderjährige: 6,50 Euro</li></ul>Gebührenfrei für<br />\n<ul
↪  class=\"list\"><li>Türkische
↪  Staatsangehörige</li><li>Asylberechtigte</li><li>
Ausländer, die im Bundesgebiet die Rechtsstellung ausländischer
↪  Flüchtlinge oder subsidiär Schutzberechtigter
↪  genießen</li><li>Resettlement-
Flüchtlinge im Sinne von § 23 Abs. 4 S. 1 AufenthG</li></ul>" ...
```

```
{...
"d115ProcessTime": "Die Fiktionsbescheinigung wird bei
↪  Vorsprache ausgestellt." ...
```

Figure 11.1: Request Body (26.04.2018)

```
{"version": "1.0",
 "session": {
  "new": true,
  "sessionId": "amzn1.echo-api.session.[unique-value]",
  "application": {
   "applicationId": "amzn1.ask.skill.[unique-value]"},
  "attributes": {
   "key": "string value"},
  "user": {
   "userId": "amzn1.ask.account.[unique-value]",
   "accessToken": "Atza|AAAAAAAA...",
   "permissions": {
   "consentToken": "ZZZZZZZ..."}}},
 "context": {
 "System": {
  "device": {
   "deviceId": "string",
   "supportedInterfaces": {
    "AudioPlayer": {}}},
  "application": {
   "applicationId": "amzn1.ask.skill.[unique-value]"},
  "user": {
   "userId": "amzn1.ask.account.[unique-value]",
   "accessToken": "Atza|AAAAAAAA...",
   "permissions": {
    "consentToken": "ZZZZZZZ..."}},
  "apiEndpoint": "https://api.amazonalexa.com",
  "apiAccessToken": "AxThk..."},
 "AudioPlayer": {
  "playerActivity": "PLAYING",
  "token": "audioplayer-token",
  "offsetInMilliseconds": 0}},
 "request": {}}
```

Can be used to generate tests in Lambda.
source: ASK DOCUMENTATION [2]

[2] ask/request-and-response-json-reference.html#request-body-syntax

List of Acronyms and Abbreviations

ACID	Atomicity, Consistency, Isolation, Durability
AI	Artificial Intelligence
ARN	Amazon Resource Name
ASCII	American Standard Code for Information Interchange
ASK	Alexa Skills Kit
AsylG	Asylgesetz, [German] Asylum Act
AVS	Alexa Voice Service
AWS	Amazon Web Serivces
B2B	business-to-business
B2C	business-to-consumer
BMG	Bundesmeldegesetz, [German] Federal Act of Registration
C5	Cloud Computing Compliance Controls Catalog
CC	Creative Commons Licence
CEO	Chief Executive Officer
CIA	Competitive Innovation Advantage
CLI	Command-Line Interface
CSS	Cascading Style Sheets
CSV	Comma-separated values
DCG	Discounted Cumulative Gain
DBMS	Database Management System
EEA	European Economic Area
EU	European Union
FAQ	Frequently Asked Question
GB	Gigabyte
GDPR	General Data Protection Regulation (EU Law)
GUI	Graphical User Interface
HCI	Human Capital Index
HTML	Hyper-Text Markup Language
IBAN	International Bank Account Number
IoT	Internet of Things
ICT	Informations and Communications Technology
ITDZ	IT Service Centre Berlin - *IT Dienstleistungszentrum Berlin*
IVR	Interactive Voice Response
JSON	JavaScript Object Notation
kbps	kilobits per second
LeiKa	Leistungskatalog der öffentlichen Verwaltung (Federal Catalogue of Public services)

ML	Machine Learning
MP3	Moving Picture Experts Group Layer-3 Audio (MPEG-3)
MVC	Model-View-Controller
MVP	Minimum Viable Product
NLP	Natural Language Processing
NLU	Natural Language Understanding
NPM	Node Package Manager
OOP	Object Oriented Programming
OS	Operating System
OSI	Online Services Index (not Open Systems Interconnection!)
PLZ	Postleitzahl (Postal Code)
ROI	Return on Investment
REST	Representational State Transfer
SLA	Service Level Agreement
SSDS	Semantische Sprachorientierte Dialogsysteme (Name of underlying Software for Virtual Citizen Assistant)
SSML	Speech Synthesis Markup Language
TF-IDF	Term Frequency - Inverted Document Frequency
TII	Telecommunication Infrastructure Index
UN	United Nations
URI	Uniform Resource Identifier
VUI	Voice User Interface

appx	approximately
app	(Mobile or web) application
capex	capital expenditure
cont'd	continued
DevOps	Development and Operations
excl.	excluding
ms	millisecond
no.	number
Inc.	incorporation
opex	operating expense, operating expenditure, operational expense, operational expenditure
rel.	relevance
ver.	version

Glossary

Access Key	9.1
Alexa	6.1
Alexa Skill	6.1
Alexa Skills Kit	6.2
Amazon Developer Console	6.2
Application ID	required by Lambda. Skill ID in our context
capital expenditure	cost of developing or providing non-consumable parts for the product or system
Intent	4.5
Interaction Model	4.5, 7.1
Lambda, AWS Lambda Function	6.2
Lex, Amazon Lex	6.2
Node.js	7.1
opex	ongoing costs for running a product, business, or system.
Polly, Amazon Polly	6.2
Public Service	1.2
Slot	4.5
Skill ID	amzn1.ask.skill.d7732837-fab2-42ff-a152-4eb0fc4ee646
Transcribe, Amazon Transcribe	6.2
Utterance	4.5
Web Service	e.g. AWS
Bot	Unless otherwise mentioned, a chatbot
Hitlist	erklärung
Voice-first	an always-on, intelligent piece of hardware, where the primary interface is voice, both input and output.