Technische Universität Berlin

DOCUMENT BUILD DATE: 22nd April 2018
DOCUMENT STATUS: Beta

Development and Evaluation of a Service Bot in 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, declare that this thesis titled, DEVELOPMENT AND EVALUATION OF A SERVICE BOT IN THE E-GOVERNMENT SECTOR and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

Erklärung der Urheberschaft

⌈ this one needs to be signed for submission ⌋

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 — Unterschrift

Berlin, den 22nd April 2018

Abstract

Though not a recent phenomenon, chatbots and voice assistants are increasingly gaining unprecedented attention as a successor for mobile and web apps. 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 on the go. They live in 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 explore the Alexa Skills Kit by Amazon in combination with respective services to develop a voice assistant for the local city council extending the current chatbot's functionality available on http://service.berlin.de. 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 in replacing traditional information lookup. We then examine the goals we define for our use case to our achievement with the APIs and SDKs available at the time. With respect to those, we also report on the opportunities, challenges and limitations faced in the development process.

Finally, we study the current state of voice assistants and service bots on the market and the future of this trend from a technical and a social point of view.

[*Supervisor*] CONCLUSION: DAS ERGEBNIS DER ARBEIT SOLLTE MAN ZUM SCHLUSS NOCH IM ABSTRACT ERGÄNZEN.]

Zusammenfassung

Auch wenn Chatbots und Sprachassistenten keine Randerscheinung mehr sind, erlangen sie in letzter Zeit eine steigende Bedeutung als Nachfolder von mobilen Anwendungen und Web Applikationen. Trotz des Mangels an definierten Standards oder feste Protokolle, mit einer zunehmenden Aufmerksamkeit auf dem Markt herrscht große Konkurrenz. Ob es durch Amazons Alexa, Apples Siri oder den Google Sprachassistenten hervorgeht, zeigen sich zahlreiche Beispiele, wie die Nutzererfahrung auf privaten oder Geschäftsebene weitgehend bereichert werden kann. Mit Lock-in Effekten spielen Drittanbieter eine große Rolle bei der Verbreitung der einen oder anderen Plattform und sorgen für neue Interaktionsmodelle. Der steigende Nutzen von Sprachassistenten zuhause, im Büro und unterwegs erlaubt viele neue Erkenntnisse und Erfahrungen, die dem Marktumfeld zu verdanken sind.

Auch im öffentlichen Dienst wird dieser Nutzen vertreten und wertvoll eingeschätzt. In der folgenden Recherche untersuchen wir Alexa von Amazon im Zusammenhang mit anderen verfügbaren Plattformen und erweitern die Fähigkeiten des Sprachassistenten durch den Webauftritt des berliner Stadtportals (BerlinOnline) um weitere Dienste sowohl für die vielfältigen und mehrsprachigen Einwohner der Hauptstadt als auch deren Senatsverwaltung. Als Grundlage wird auf die Funktionalität des Chatbots auf dem Stadtportal http://service.berlin.de 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. Anschließend werden die erreichten Ziele mit den gesetzten Anforderungen in Perspektive gestellt. Darüber hinaus wird über Systemgrenzen sowie individuellen und systematischen Fehlerquellen berichtet, die bei der Entwicklung behandelt worden sind.

Schließleich nehmen wir den aktuellen Trend von Sprachassistenten und Chatbot-Dienste unter der Lupe und diskutieren eventuelle Zukunftentwicklung mit dem gegeben Stand jenseits des e-Governance Bereichs. **[*Supervisor*] CONCLUSION: ERGEBNIS SOLLTE ZUM SCHLUSS NOCH IM ABSTRACT ERGÄNZT WERDEN.]**

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 [20] and a substantial fraction thereof using smarthome gadgets and appliances on a daily basis, AI's role 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 and Natural Language Processing 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 [14]. 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 [22]. 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 [19], 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% [10] [21] of all searches by 2020 with currently highest rates coming from the youngest generations based on a survey [23] 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 [13], 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 [5]. 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 [15] 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) [7]. 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 [8], 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 [17]

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* [5] 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 5.4 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 3)

1.2 The Analyzed Scenario

Berlin.de is an online one-stop-shop for approximately 3,7 million residents of the German capital [4] with ⌈ hundreds/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 [12] 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 [11].

1.3 Approach and Goals

In the following thesis, we take these factors into consideration and narrow them down to fit our tailored scenario in the e-Government sector. We start by surveying industry-standard voice assistants platforms like Siri, Bixby, etc. and frameworks such as Microsoft Azure Bot Framework, Amazon Lex and API.ai, and present a solution that caters for the local municipality of Berlin, Germany.

For this we choose to present our solution using Alexa based on our criteria in section 2.4.

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" and analyse the strengths and weaknesses of choosing this platform. We start by exploring the available public services on https://service.berlin.de to understand how to implement our Skill. We then choose a set of examples with different levels of implementation difficulty to prototype how all current and future Berlin.de's services should be handled using knowledge from existing databases. Difficulty is expressed in the required knowledge of the Skill to perform a user's request. ⌈ This includes for instance abilities like the system to understand the district location intended by the user through a postal code input. ⌋

Further, we test the Skill using unit tests for the code and usability tests for the interaction model especially with the ability to identify and deal with. In the code, we alread have the advantage of many small-talk abilities since we build on top of Alexa's existing and continually expanding database for question-answer models. ⌈ mention that this could get you out of the skill in the pro/con section. ⌋

In summary, these goals were defined for the system:

- understand user's requests about public services through different sentence formulations
- deliver relevant information based on the service requested
- communicate with an endpoint that builds on the service catalogue of Berlin.de's "Virtueller Service Assistent"
- produce a scalable, modular system that can be extended in the future

Further requirements are discussed in Chapter 3.

1.4 Structure of the Thesis

This thesis endeavours to shed light on the following: In Chapter 2, we first introduce the Alexa platform and the idea behind Skills then discuss related work as currently available. Going beyond related work in the field, we show a few comparative implementations for a chatbot and a voice assistant For that we consider **the City of Vienna chatbot service "WienBot"** given that it also uses German and the Alexa Skill **"AskGeorgia"** for a comparison in English. We compare them to the current chatbot application "Virtueller Service Assistent" available on service.berlin.de, which we build our work on with the same underlying technology using Apache Solr.

In chapter 3 we introduce our design to the Skill and discuss how it can overcome a few struggles with respect to redundant boilerplate code, smalltalk abilities and briefly highlight some limitations on the system limitations. We also present the frameworks and Application Programming Interfaces (APIs) we use in our own implementation, the prerequisites and the artefacts provided initially through a breakdown into the following ⌈ 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)"

Chapter 4 represents a detailed analysis of our implementation solution. In particular, 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 code is available on GitHub [1] with a digital copy attached to this work.

Chapter 5 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 6, we conclude with how the implementation of this e-Government solution is attainable through Alexa and where future works can be directed.

[1] `https://www.github.com/megantosh/AlexaImAmt`

✍ **TO-DO**: make sure this is final URL and is public before printing

Chapter 2

Background

In this chapter we discuss Amazon's state-of-the-art strategy before moving on to related work in other archetypes as it is important to introduce the implementation scope of voice assistants bottom-up prior to exploring the current context within the same boundaries of voice assistants then compare it to other approaches in the larger context of conversational bots as a whole from a technical and user experience point of view. We start with a juxtaposition of 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.

2.1 GUI vs. VUI

Visuals and Sounds can both communicate the same message even though the medium used is completely different. 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 image to the receiver. Communication through voice, on the other hand, can generally allow more room for interpretation. We think of an example where you would watch a football game on TV vs. the same game on radio, 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 receivers 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 ground for understanding and the short written text becomes purposefully designed to give in the best case a message not open to interpretation and so a foundation for an agreement or contract between the source and the destination of the message. It is a good medium of reference.

The utility of Voice comes handy with a different paradigm in numerous settings.

Say you are at an airport and want to catch your flight, or rushing to catch a meeting. Would it not be more effective to ask someone on the information you need quickly if you can ask it in a precise sentence quicker than you would write it, or visually navigate on a web page / app till you reach the information you want? 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, for instance having to look on a screen. As our ambitions in connectivity become more complex, we have become more dependent on our phones in the last decade. This is where use of a Voice User Interface could have the advantage of liberate us from the constant usage of our smartphones and moving from one screen to another. However, it is noteworthy that the trade-off between information exchange visually and through audio 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 is an important factor.

Hence, we need to differentiate between GUI design an that of a VUI. Since we come froma GUI background thanks to the World Wide Web standards with HTML and CSS, 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.

✍ **TO-DO**: If this needs more explanation, start a sentence like this: "Imagine you have an intent resembling the label on th button for "ok". with a function linked to it to perform a commit action, ..."

✍ **TO-DO**: It's important to understand that we are at the beginnings of VUI design, but are making progress, e.g. VUIs - should become seamless in an approach like this: Amazon and Microsoft agree their voice assistants will talk (to each other)
- additional plugins are in the making, giving individual touch of sound based on profile (Alexa Podcast on the way to BXL)
- art of bot design
- best practices

⌈ is this okay here or should it go in an appendix? ⌋

✍ **TO-DO**: - **Why can't robots understand us:** language ambiguities - the need to understand context
–Syntactical: Homonyme
–Semantic: Methaphors, sarcasm, and puns
–dialects: enunciations
–underlying grammar
–underlying sentiment
-**NLP Progress:** How does it help in enriching the bot experience
–neural networks: help understanding language patterns and get better over time
–thought vectors: helps connect different words with related meaingns
— link: fortschritt, und systemgrenzen heutzutage (kurze Sätze etc)

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 3.3.

Having the interface disappear is therefore a game changer for as soon as the users 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 voice, the user is uninhibited in what they say and the system has to understand the *intention*of the user with *the formulation*they just used. For us to hold on to that sentence, we need to define the lexis we use for voice design. Just like most common chat bot constructs, Alexa Skills Kit (ASK) divides the building model into intents, utterancesand slots The Fulfilment part is taken care of through a back-end endpoint containing the programming and business logic to the interface, which also validate the slots before sending it onwards, similar to what JavaScript checks would do on a website. We make the following words clear:

2.1.1 Terminology

intent 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 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 a variable fo 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 *big* coffee, they assume that there is the option to have a small coffee, too. And so big and small refer both to the size of the coffee. Programmatically, we define `big` 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:

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

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 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.

With these introductory definitions we try to lay out a foundation for a standard, which is still in an experimental phase. Much like when tablets and smartphones were introduced to the market, the languages and framework used to render GUIs for the users had to adapt, e.g. with HTML5 and CSS3. In that transition, the experimental phase can be considered when browsers had to render a different page depending on the client. It started with a certain few screen resolutions that can be fixed, to offering a desktop and a mobile version and finally to the spreading of responsive design, the MPEG4 H.264 codec for video and so forth. When screen sizes and aspect ratios became also so diverse that it was hard to keep track of, website designers saw the importance of using different relative measuring units to present the website correctly (e.g. percentages for CSS classes, ems for fonts, etc.)

Since we mention context, in voice design we notice that the essence of an implementation lies in being able to think with a user mindset. In that respect, the developer of a voice design automatically becomes a designer to some extent. Part of that process is to understand where and when is the user going to request a certain intent. This helps us figure out how to offer the best user experience. For instance, it can be sometimes more effective to integrate the GUI inside the VUI, where the user gets to see relevant images, text or other elements related to the spoken responses by the visual assistant.

As we present our solution in the Alexa environment, we would like to familiarize ourselves with further terms that apply specifically to that platform. This is a prerequisite to understand the script building the interaction model with the user and helps us understand the code templates provided throughout the development process. For more information, please visit the Alexa Skill Design Guide at
`https://developer.amazon.com/docs/custom-skills/create-the-interaction-model-for-your-skill.html`
and
`https://developer.amazon.com/docs/smapi/interaction-model-schema.html` We cover here an excerpt of basic elements that we will later see more or less in the same structure of the JSON file containing the interaction model:

✍ **TO-DO**: explain json (this and LeiKa's) zu Ende
mention Alexa Voice Service and Products

✍ **TO-DO**: should I introduce concepts like entity resolution here already?

Table 2.1: Interaction Model Schema (Skill Management API) [9]

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			

2.2 State of the Art

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 [10]. As throughout the course of this thesis these major changes occurred, we will go over the workarounds and circumventions to retain a working version of the software we present, discuss

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. [18] 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

Alexa is a cloud-based voice service assistant platform by Amazon powering millions of devices ⌈ with {number of requests} daily—monthly ⌋ on its own-branded 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.

Alexa's AWS Modules ⌈ make logos as minipage objects or set as table / problem with footnotes ⌋

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" [18]. While 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 voice assistants and conversational bots independent from Alexa, Siri etc., development for Lex was not available in German at the time of this research. We therefore decide to use the Alexa Skills Kit, which take advantage of Lex and

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

[2] see Etymology in Annex 6.3

the other components described below. Lex and Alexa use the same deep learning techniques for natural language processing with the workflow described with the example in figure 2.1. For more information on Lex and its difference to Alexa, please refer to Annex 6.3.

Polly [4] *for speech synthesis*
another "service turn[ing] text into lifelike speech, allowing [developers] to create applications that talk" [18] 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" [18]. 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.

✍ **To-do**: don't mention I'm using lambda until design chapter 3

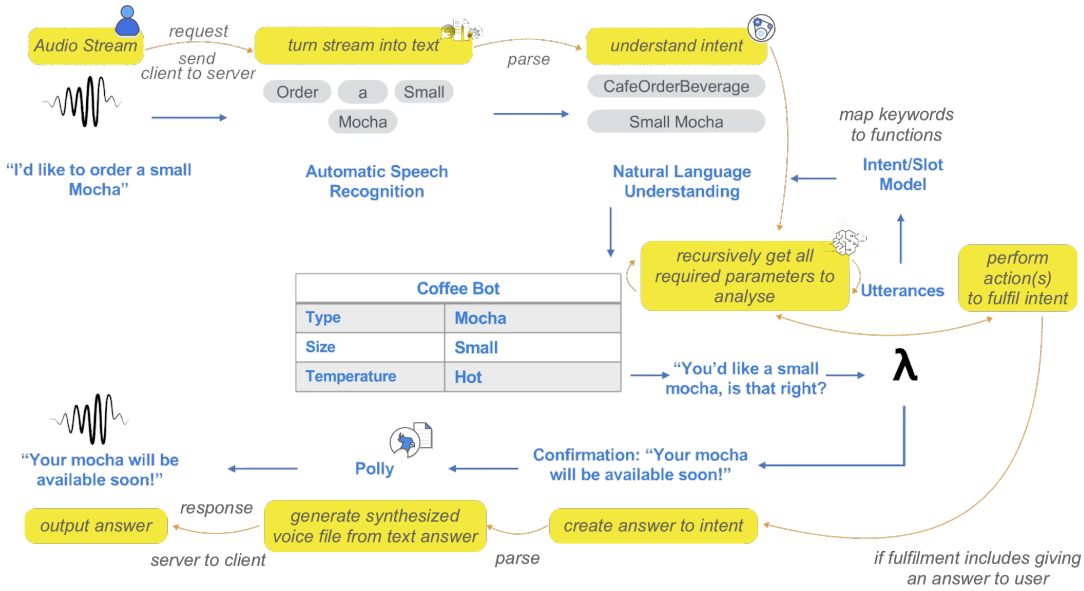
Lambda [6] *for intent fulfilment*
Although Lambda is a versatile "service [built] for a variety of real-time serverless data processing systems" [18], ⌈ we use it / it can be used ⌋ as an integrated server instance to host our back-end code for intent fulfilmentWe might as well use just any other privately 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 program we develop (an Alexa Skill as we describe below).

CloudWatch [7] *for event logging*
CloudWatch acts like the console for an operating system. 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.

IAM [8] *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 [9] *for identity access management beyond AWS*
like the rest of AWS's platform, Cognito is a scalable service module to perform user authentication and can be used in combination with an external endpoint instead of Lambda.

Figure 2.1: Interaction between AWS modules in the use case of a "Coffee Bot" based on reference [3]



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 below for a use case of a hypothetical chatbot that operates a coffee machine using some of these service modules (Figure 2.1). 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 extention of its aggregator-model marketplace model by playing as a mediator between the developers' role and that of end-users

we can therefore describe the meta-model for Alexa similar as one of an application with several front-end and back-end components. Unlike in most GUI-based scenarios with an MVC design pattern [6], 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. These are broken down into the following elements for users and developers to comprise a holistic ecosystem:

Alexa Skills Kit 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 makes up a good representation of the JSON Files that include the language (interaction) models, the skill properties, the endpoints it uses and is where to submit the skill for publication

ASK CLI `ask-cli` A Command-Line Interface tool that interacts with the Alexa Console skipping the web browser. It is still in the making but has numerous functions to creating, deploying and testing our Skills. Download with Node Package Manager: `npm install ask-cli`

Amazon Voice Service ⌈ products independent from Alexa, not sure if it's worth mentioning ⌋

Alexa Skills Store where the end-user can preview the Skills before installing them, Once installed, the own instance of Alexa becomes "smarter" by that Skill, which does not need

to update from client side, since it is not hosted on the client (only sends requests to it). [10] for now the user does not need to think about updates since these happen in the back-end.

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**: remove clearpage before print

[10] 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 anymore 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.

Alexa Interfaces

Hardware

Table 2.2: Alexa Devices in Comparision

category	Speaker	Tablet	SmartHome	TV
curr. 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	E. Show: Bluetooth Kindle: yes	yes	via HDMI
Alexa launch	voice cmd	K. 7, - HD 8: btn press - HD 10: voice cmd	voice cmd	btn press

Software

The Alexa app is not an Alexa interface yet (although this will change soon).

while the hardware models stated above are possibilities for testing, too, they do not primarily serve an optimal testing environment. There are better 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 conversations retaining sessions, so that one does not need to repeat the full conversation until one reaches the test breakpoint.

EchoSim.io “an online community tool for developers that simulates the look and feel of an Amazon Echo” [11]

Alexa Simulator an online simulator giving JSON responses and voice feedback to the requests sent. We interact with it either through voice or with JSON requests. Also available through the ASK CLI.

Reverb An iPhone / Android app, allowing the interaction with the Alexa instance linked to an Amazon account. [12]

[11] `https://echosim.io/`
[12] `https:reverb.ai`

2.3 Related Work

✍ **TO-DO**:
eingehen auf
- Wienbot
- Singapore / LA / Ask Georgia...
- LeiKa
While we have no access to AskGeorgia from the German Amazon Store, we use the US store.

✍ **TO-DO**: move / remove what is irrelevant In this section, we conduct a short survey of problems we can face with natural language

topology of bots

by platform: -API.ai / Facebook Messenger Chatbots / -wit.ai / -motion.ai / Flask **by category**
- leisure / - fun bots / - productivity / - more (graph from voicelabs report)
- what are classic use cases for their use with prominent examples?
Booking tickets (e.g. airline bot)
- quick survey of respective 'AppStores'
by purpose
-physical locations (home, office, car, phone, in a business)
Information bots
- mention available service types (information system as a "webpage/database")
- vs an interactive bot that gives you customized information on demand hier soll der D115 Anwendungsfall "Beauskunftung" kurz erläutert werden
social bots
- with advantages / disadvantages
- fake news / online reviews
more on AI in bots (optional)
- use of ML
Handyversicherungsbeispiel
- from business perspective, the bot is aiming to sell more polices,
- the bot tries to determine if there is a nuance in the user's answer (machine acting as a judge!) - e.g. "how did the phone fall off" - MKTG - Aufwand IFTTT Applets for voice commands

✍ **TO-DO**: 2 ¶

- Chatbot vs. human: Analyze differences between bot and human response -disadvantage: a bot wants a sentence broken down in small pieces to avoid errors in lengthy interpretation
- **wrap-up:** can bots replace serivces offered by humans? – mention transition from facets (Altavista) to metasearches to all-in-one (Google).
– chatbots as enablers in customer service industry
– conclusion: Although not impossible, it is a bit too far-fetched at this stage.

✍ **TO-DO**: - structure of Hitlist on berlin.de is provided by ITDZ - as opposed to Versicherungsfirma z.B (ML tries to detect irregular patterns in case customer is lying). - unfortunately forums vs. FAQs did not work. if i want assistance, i want the customer to tell me the model number - and forums have mostly Schrott!
what the bot curently achieved is at least not give wrong answers, sometimes says idk but it doesnt confuse u. same attitude like in lmny shops (nur unpassende antworten sind frustrierend!

2.4 Choice of Platform

✍ **TO-DO**: here you can talk about alternatives like Microsoft, etc, compare **pricing scheme**, say why you chose alexa (for its popularity mainly and 3ashan Api.ai et3amal abl keda
w 3ashan it's a whole established ecosystem. ma3 el ta7afozat eno amazon has an obscure position to data and privacy vs. siri masalan.
3ashan ma3 siri me7tag iOS knowledge

w ba2a 7aga like Maintain context(sesions), Intent chaining (khodlak kam screenshot men Azure)

- on top of that, alexa does this cool thing where it gives

✍ **TO-DO**:
Intent fulfilment - mention how ability to react to everything is centralized 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

2.5 The Analysed Scenario as a Use Case

D115 and The Currently Deployed "Virtual Service Assistant"

✍ **TO-DO**: - summarize infobroschuere_ BMI08324_screen_barrierefrei.pdf
-Use case im Detail
-Welche Daten gibt es?
-Was sind die Erwartungen?
- wie kann man die Güte des Systems beurteilen? **do not forget the surveyu made**
- Meist sollte man in diesem Kapitel die Lösung schon im Auge haben, um die Erwartungen so zu formulieren, dass die Lösung auch geeignet ist?

✍ **TO-DO**: **current berlin.de bot**
- dienstleistungen.json structure (finding the info through hierarchical nodes)
- interpreting the nodes as intents
- traversing the nodes (one level up then to next node)
- no session/no persistence
- explain how the json nodes map to intents and cores in solr etc

provided in JSON for value lookup, the structure of the file is as follows: there are ⌈ carry on - the structure ⌋
started with 616 Intents in `data` node (now 685 or so), each containing ⌈ ⌋

✍ **TO-DO**: missing variables e.g. are required papers, flag: persönliche Vorsprache ja nein, ...

✍ **TO-DO**: change this into a table and add a tree list like the interaction model in appendix

Figure 2.2: `Dienstleistungen.json` - Primary Nodes

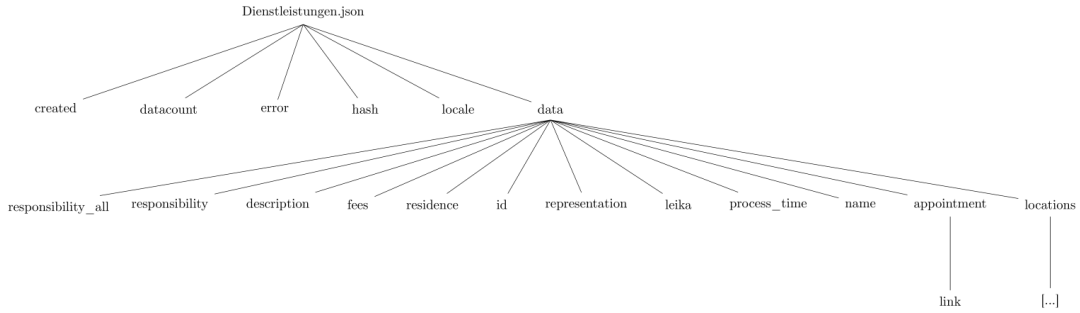


Figure 2.3: `Dienstleistungen.json` - secondary Nodes

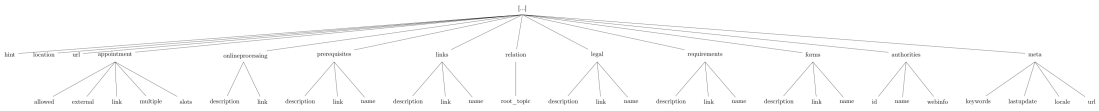


Table 2.3: relevant nodes in query results `Dienstleistungen.json` - Description

key	type	ex. Value	descriptio
`id`	`int`	326233	public ser
`d115URL`	URL `string`	”https://service.berlin.de/dienstleistung/326233/”	link to pu
`d115Name`	`string`	Fiktionsbescheinigung	public ser
`ssdsAll`	`string`	...	additional
`d115Description`	`string`	...	Introducto

Chapter 3

Skill Design

✍ **To-do**: intro on structure of chapter
- to then elaborate on implementation requirements in section 3.1. - 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)

✍ **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/`

3.1 Frameworks and Data Structures

Node.js

AWS Lambda supports multiple runtime environments including Python, Java, C# and Go. 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.

✍ **TO-DO**: - 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

Apache Solr

w lucene w tika w nutch wel habal da if required
using state of the art standards in TF/IDF for seach queries
-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

Setup in Java using Maven [1]

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

Additional APIs

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

- swagger

- Flask

✍ **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-4

- 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

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

Description of the Interaction

✍ **TO-DO**: seqence diagram as found in -inbox-

3.2 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
- wo hilft mir alexa, was mach ich lieber woanders?
- Ähnlichkeitsmaße -levenstein-distanz, IFTTT

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 ”?”

3.3 Design Guidelines and Documentations

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

3.4 Best Practices

Throughout the model building process, we come across various nuances and details that might look subtle from a programming point of view. However, they very much can enrich or completely spoil the user experience, resulting in the users not returning to use our Skill again and can be an acceleration towards 1-Star reviews.

Based on our own exploration and through a growing voice design guide offered to Alexa developers [2] as well as personal recommendation from Alexa evangelist Memo Döring [16], 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

[2] `https://alexa.design/guide`

1. **Design before Implementation**
   Since the Skill code usually starts small, it might be tempting to just 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 would 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. Therefore, it is for every developer to figure out how to move to the VUI paradigm step by step with regard to what is important to them.

2. **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 right pattern, which eventually make us speak a language. There are many rules regarding shapes, abstract concepts. When we build programmatically, it is possible to get lost in the order of words in a sentence or even in the order of sentences.

Of course 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.

3. **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.

4. **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:

   "I want to bake a cake"

   Alexa should respond with something like

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

   then when we get back to Alexa an hour later, 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.

5. **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.

6. **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.

7. **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.

8. **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 "

   This 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

9. **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.

10. **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 [3] and English [4]

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

" First, ...", "Second", "At the end"

11. **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 ..."*

[3] https://developer.amazon.com/docs/custom-skills/speechcon-reference-interjections-english-us.html

[4] https://developer.amazon.com/docs/custom-skills/speechcon-reference-interjections-german.html

12. **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.

13. 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.

14. **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.

15. **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.

16. **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 4). Checking when and how a Skill fails is of great value to enhancing it.

17. Keeping it Simple

    A Skill that uses 1000 intents is more likely to break. If we can break these down into more modular intents, we avoid the risk. Same goes for utterances. 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.

18. 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 they are random to Alexa. E.g. 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 give them a joke. Since the user was expecting to challenge or break the Skill, they would get surprised if they end up learning about something new or have a laugh instead.

Chapter 4

Skill Implementation

4.1 Setup

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

4.2 Code Analysis

4.3 Deployment

4.4 Approval and Publishing

4.5 Debugging / Troubleshooting

4.6 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

Chapter 5

Evaluation

5.1 Alexa in General

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

5.2 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 `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

5.3 Results

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

5.4 Discussions

- 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

Chapter 6

Conclusion and Future Work

6.1 Summary

6.2 Conclusion

6.3 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)

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.

-matensash el evaluation beta3et el chatbot

Bibliography

[1] Adam Marchick, Alpine.ai. Voice search trends. https://alpine.ai/voice-search-trends/, Apr 2018. 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. Accessed on 16.02.2018.

[3] 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. Accessed on 15.04.2018.

[4] 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. Accessed on 06.02.2018.

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

[6] Wikipedia contributors. Model-view-controller. https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller.

[7] 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. Accessed on 14.02.2018.

[8] 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. Accessed on 14.02.2018.

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

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

[11] Georg Berger, RBB Online. Geniale köpfe - wohnen der zukunft. https://rbb-online.de/abenschau/archiv/20180416_1930/sahin-Albayrak.html, April 2018. Accessed on 16.14.2018.

[12] 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.

[13] Tobias Grossmann, Regine Oberecker, Stefan Koch, and Angela D Friederici. The developmental origins of voice processing in the human brain. *Elsevier*, 65:852–8, 03 2010.

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

[15] 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. Accessed on 17.02.2018.

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

[17] 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. Accessed on 15.02.2018.

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

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

[20] 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. Accessed on 06.02.2018.

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

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

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

Appendices

Annex: On Etymology

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.

✍ **TO-DO**: did something go missing here? check git!

Difference Between Lex and Alexa Skills

✍ **TO-DO**: cite? indent? copied text

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 ćhatbotsśo 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 MobileHub and Amazon CloudWatch and easy integration with many other services on the AWS platform including Amazon Cognito, and Amazon DynamoDB makes bot development effortless. For 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/

Outtakes from a JSON query node

```
{
"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>Unterlagen fehlen oder die
 → Ausländerakte nicht vorliegt,</li>
[...]
```

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

```
"d115InfoLaw": ["§ 81 Aufenthaltsgesetz - AufenthG
::: false ::: http://www.gesetze-im-internet.de/
aufenthg_2004/__81.html"],
```

```
"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\ ::: ",
"Gültiger Pass oder Passersatz ::: Eine
 → Fiktionsbescheinigung gilt immer nur in Verbindung
 → mit einem gültigen Pass oder Passersatz. ::: "
]
```

```
"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."
```

Appendix A: Abbreviations

AWS	Amazon Web Serivces
ASK	Alexa Skills Kit
AVS	Alexa Voice Service
ARN	Amazon Resource Name
MVP	Minimum Viable Product
CLI	Command-Line Interface
MVC	Model-View-Controller
CIA	Competitive Innovation Advantage
IVR	Interactive Voice Response
SSML	Speech Synthesis Markup Language
AI	Artificial Intelligence
NLP	Natural Language Processing
ML	Machine Learning
GUI	Graphical User Interface
VUI	Voice User Interface
ACID	Atomicity, Identity, .. criteria
appx	approximately

Appendix B: Glossary

Intent	erklärung
Slot	erklärung
Utterance	erklärung
Alexa	erklärung
Alexa Skill	erklärung
Alexa Skills Kit	erklärung
Amazon Developer Console	erklärung
Lambda, AWS Lambda	erklärung
Lambda Function	erklärung
Lex, Amazon Lex	erklärung
Polly, Amazon Polly	erklärung
Amazon Transcribe	erklärung
ElasticSearch	erklärung
Node.js	Framework built on top of JavaScript
Interaction Model	erklärung
Service	bot, AWS, Berlin.de
app	Mobile Application
https://docs.aws.amazon.com/general/latest/gr/glos-chap.html	
Application ID	erklärung
Skill ID	erklärung
Bot	Unless otherwise mentioned, yeb2a Cha
Hitlist	erklärung
Voice-first	an always-on, intelligent piece of hardw
B2B	business-to-business
B2C	business-to-consumer
Inc.	incorporation