

Sonja Merisalo

Developing a Chatbot for Customer Service to Metropolia UAS Student Affairs Office

Helsinki Metropolia University of Applied Sciences

Master of Engineering

Information Technology

Master's Thesis

26 September 2018

PREFACE

Chatbots hit me in the face around November 2016 when a colleague of mine told me (with huge disbelief) that she had just talked with a chatbot - and thought that it was a real person! Of course, I had to try it out and it was quite impressive. The urge to find out more stayed in the back of my head and spurted out when I got into master-studies. The hardest part was how much to narrow down. At first, I thought I could actually make a real, working chatbot with Natural Language Processing (NLP) and high-level Artificial Intelligence (AI) - including all the testing with actual users, iterating, launching, marketing, piloting period, collecting feedback... Yes, that would have been a doctoral thesis. Since as usual, my goals and expectations were at least sky high. Ending up with “just” a proof of concept was not ideal for the ambitious me, but it was the only reasonable thing to do. Anyway, oh boy how much I learned about chatbots, Machine Learning, NLP, frameworks, libraries and coding.

Thank you, my instructors! Erja Nikunen for jumping in and giving the most important technical instructions and also pulling my head down from the clouds, and Harri Airaksinen and Ville Jääskeläinen for all those inspiring and down-to-earth thoughts. Thank you, my persistent tech-angel Fredrik Gundersen for getting me started with the RiveScript framework and Teemu Vilkmán & Ville Raassina for giving some nice technical and designing hints. Thank you, my dear beloved husband for carrying the load of two parents every once in a while. Thank you, my dearest, my three darlings, who let mommy study “a bit” and gave outspoken feedback of the chatbot in the making.

Sälinkää, 18 September 2018

Sonja Merisalo

Author Title	Sonja Merisalo Developing a Chatbot for Customer Service to Metropolia UAS Student Affairs Office
Number of Pages Date	52 pages 26 September 2018
Degree	Master of Engineering
Degree Programme	Information Technology
Specialisation option	Networking and Services
Instructor(s)	Erja Nikunen, Principal Lecturer Harri Airaksinen, Principal Lecturer
<p>This thesis is a study about chatbots, including a proof of concept about a chatbot planned to the use of Metropolia UAS Student Affairs Office (client). In the background is a real-life problem: how to serve students fast and automatically, and also reduce customer servants' routine answering to students' basic questions. In this thesis the history and theory of chatbots is presented, and a short comparison of some chatbot solutions. However, the main focus is on how to design a chatbot as a whole and creating of a demo, e.g. proofing the concept. Usability and testing are also shortly presented, as well as recommendations and alternative options for the technical implementation and how to develop the demo further.</p> <p>In the demo a ready framework was used in order to fit into a scope of a thesis. RiveScript turned out to suit best for the client's requirements of Finnish as operating language, free of charge, directing questions to customer servants and answer including a link. The chatbot was decided to be put to Metropolia's intra, to Student's Desktop, rising a requirement of chatbot operating from Metropolia's servers.</p> <p>NLP and anything more than basic weak AI were left out of the demo, since they weren't mandatory for the demo to work. However, at least NLP should be added to the actual chatbot, since then the usage is more efficient, especially from the Finnish word forming's point of view, and then the chatbot would be easier to build more conversational.</p>	
Keywords	Chatbot, Machine Learning, Artificial Intelligence, NLP

Kirjoittaja Otsikko	Sonja Merisalo Developing a Chatbot for Customer Service to Metropolia UAS Student Affairs Office
Sivumäärä Päivämäärä	52 sivua 26 syyskuuta 2018
Tutkintotaso	Master of Engineering
Tutkinto-ohjelma	Information Technology
Suuntautumisvaihtoehto	Networking and Services
Ohjaajat	Erja Nikunen, Yliopettaja Harri Airaksinen, Yliopettaja
<p>Tämä insinöörityö on tutkimus chattiboteista, sisältäen proof of conceptin Metropolia Ammattikorkeakoulun Opintotoimistolle (asiakas). Taustalla on aito ongelma: kuinka palvella opiskelijoita nopeasti ja automaattisesti sekä samalla vähentää rutiininomaista vastaamista opiskelijoiden peruskysymyksiin. Tässä työssä esitellään chattibottien historiaa ja teoriaa, sekä lyhyt vertailu muutamista chattibottiratkaisuista. Pääroolissa on kuitenkin chattibotin kokonaisvaltainen suunnittelu sekä demon, eli proof of conceptin, rakentaminen. Myös käytettävyys ja testaus käydään lyhyesti läpi sekä annetaan suosituksia teknisen toteutuksen ja sen kehittämisen suhteen - sisältäen vaihtoehtoisia ratkaisuja.</p> <p>Demo rakennettiin valmista pohjaa käyttäen, jotta insinöörityön laajuus ei ylittyisi. RiveScript sopi parhaiten asiakkaan vaatimuksiin: suomen kielen käyttöön asiakasrajapinnassa, maksuttomuuteen, kysymysten välittämiseen asiakaspalvelijoille sekä linkin lisäämisestä vastaukseen. Chattibotti päätettiin sijoittaa Metropolian intraan, opiskelijan työpöydälle, joten yhdeksi vaatimukseksi nousi myös sijoittuminen Metropolian palvelimille.</p> <p>NLP ja muu kuin alkeellinen heikko-AI jätettiin demon ulkopuolelle, sillä ne eivät olleet välttämättömiä demon onnistumisen kannalta. Vähintäänkin NLP olisi kuitenkin syytä lisätä chattibottiin, koska sen avulla on helpompi hallita suomen kieltä ja tehdä keskustelusta chattibotin kanssa aidomman oloinen.</p>	
Avainsanat	Chattibotti, Koneoppiminen, Tekoäly, NLP

Table of Contents

Preface

Abstract

Abstrakti

List of Abbreviations and Key Terms

1	Introduction	1
2	Method and Material	3
2.1	Background of this Thesis	3
2.2	Research Design and Solution Development	4
2.3	Solution Evaluation, Reliability and Validity	5
3	Theoretical Background	6
3.1	History of Chatbots	6
3.2	Chatbots and AI	8
3.2.1	Chatbots in Dialogue	10
3.2.2	From Basic Bots to Smart Bots	10
3.2.3	Chatbot Trends and Flip Sides	14
3.3	Creating a Chatbot	14
3.3.1	Designing a Chatbot	15
3.3.2	Building a Chatbot	17
3.3.3	Comparison of Possible Chatbot Solutions	18
3.3.4	RiveScript as a Solution for Chatbot	19
3.4	Usability, Testing and Releasing	24
4	Results and Analysis	29
4.1	Defining Requirements for the Chatbot	29
4.2	Designing the Chatbot	30
4.3	Creating the Chatbot	35
4.4	Future Development	39
4.5	Alternative Technical Solutions	41
4.5.1	GetJenny	41
4.5.2	Giosg	42
5	Discussions and Conclusions	43
6	Summary	45

References

List of Abbreviations and Key Terms

AI	Artificial Intelligence
B2B	Business-to-business
B2C	Business-to-customer
Chatbot	A conversational agent
FAQ	Frequently Asked Questions
HCI	Human-Computer Interaction
IA	Information Architecture
IM	Instant Messaging
Machine Learning	A field of computer science giving the computers the ability to learn with data without specific programming
Neural Networks	e.g. Deep Learning, an algorithm trying to simulate neurons working in brains
NLP	Natural Language Processing
PaaS	Platform as a Service
SaaS	Software as a Service
Strong AI	A machine or program with sapience and logical reasoning abilities, consciousness
UAS	University of Applied Sciences
UI	User Interface
UX	User Experience
Weak AI	A machine or program capable of pattern matching and other narrow function

1 Introduction

This thesis is a study about chatbots: what they are, how to design one, what kind of solutions can be used and how to create a simple demo with a ready framework. Why to study this topic? Simply because chatbots are those neat, cost effective solutions, which are taking over the traditional chats, formerly run by humans. The reasoning to this happening is quite simple: what is the original idea of a chat? Talking with people! However, lately chats have been used more and more also in customer service. What is then the idea of a customer service? Helping people! And people tend to need help for quite similar issues inside of a certain web service. What do the customer servants do in some extent? Copy-paste the answers from a database! And that is something a bot can do also, just clearly faster. Therefore, the idea of a chatbot is to give the users fast, informative and cozy customer service experience - and to release the human resource to do something more creative.

Chatbots have emerged in the last few years and are now finding their way to education too. In USA some Universities use Artificial Intelligence (AI) and/or chatbots in their services. More about this in the Discussions and Conclusions -chapter. In Finland, this is still very new, and that was one more good reason why chatbots became the topic of this thesis. A chatbot can be seen as a tool to increase sales for example in airlines, mobile operators or in any kind of web shop. A bit less frequently, the pure goal is to help people, without having any revenue expectations in the background. To the latter category goes the chatbot focused in this thesis: a chatbot to customer service in University of Applied Sciences (UAS). Chatbots aren't so frequent topic to study in the field of education and especially in university level. This thesis is one of the few in Finland.

Metropolia University of Applied Sciences is the biggest UAS in Finland. It has approximately 16700 students in 67 degree programmes in the fields of Business, Culture, Health Care and Social Services and Technology. The values of Metropolia UAS are High quality, Expertise, Transparency and Community spirit. Using a chatbot in customer service brings transparency to the operating policy and might even strengthen the community spirit.

Metropolias' Student Affairs Office (client) is a team of approximately ten people handling all the study-related issues Metropolias' students may have. A typical way the students

contact the client is via email and the common things students ask tend to pile up in certain topics. The customer servants answer even with using copy-paste answers, since the answers are almost identical. Because everything must be super-efficient nowadays, the head of the Students Affairs Office wanted something to help the customer servants in their daily work, to release resources from the most routine tasks. So how about a chatbot, which would take care of the most routine answering to the students' questions?

The client targeted the chatbot to students who already are in Metropolia - another chatbot was hoped to be created to admission services in the future. Metropolia has an intranet, OMA, which provides desktops to certain roles, for example Student's Desktop. What would be more convenient than having a chatbot in that Student's Desktop to help Metropolia's students?

This research, which is based on theory and existing knowledge, is about designing a simple chatbot and making a demo: creating a proof of concept. It was clear to use a ready-made solution (a platform or a framework) since coding a chatbot from scratch takes months and requires a lot of coding skills. Most of the work with the solution considered adapting and configuring, modifying and cultivating. This thesis shows the process of choosing and implementing: which of the existing solutions suits best for this precise need? How to design a chatbot to UAS? Some unstructured interviews and a lot of googling were made to find information about the different approaches and choices.

The actual outcome of this thesis is a proof of concept. From choosing a framework and designing a chatbot which fits to the client's needs to creating a demo and offering recommendations how to create the actual chatbot. Since this study is "just" a Master's thesis, Machine Learning, NLP and anything more than basic weak AI had to be excluded (when a chatbot is based on a static database and enables simple learning it can be seen as basic weak AI). Adding at least NLP is set as a target to the actual chatbot.

This thesis is divided into six sections. The first introduces the problem, the second presents the methods and materials. The third section sheds light to the theoretical background, the fourth is about the actual making and finally the fifth points out the conclusions. The last section, number six, summarizes all up.

2 Method and Material

When planning and creating a chatbot, the theory lays in software development, Human-Computer Interaction (HCI), Machine Learning and AI. One also needs to understand the usability and psychology behind usability. There is no point of planning any service to humans without thinking of the humans. Usability should be number one priority. This leads to the need for usability testing, which is something no designer should ever neglect. In chapter three these theories are presented more precisely, in this chapter the background of the project and the solution are described.

2.1 Background of this Thesis

As told in the Preface and Introduction, the inspiration towards chatbots lays both in personal interest of the author and in actual need. The author contacted a few potential clients inside Metropolia, but the highest impact to student's life was found in the needs of Metropolia's Student Affairs Office. Finding ways to support the students in their studies is important to the author. Metropolia Student Affairs Office wanted to get rid of some routine emailing in order to reduce the workload of their personnel and also to serve their customers (students) better.

The chatbot demo created in this thesis had at first hopes and waits of having learning capabilities and being able to be more conversational, but the reality turned out different. Since this is just a thesis, the scope had to be narrowed down to a basic chatbot: no more than basic weak AI, no NLP or such. This level of demo is enough for time being, since the most important thing was to proof that a chatbot can give the correct information, and that the chatbot can do so with simple sentences (including umlauts) and links. The suggestions for future development presented in this thesis for the actual chatbot include adding at least NLP to it, perhaps also more AI / Machine Learning in the form of deeper learning, since they are needed for more efficient use of Finnish and they make the interaction more conversational. In later development, English as operating language option is wanted and even later the client hopes to have another chatbot: one which serves the people applying to Metropolia.

2.2 Research Design and Solution Development

This thesis includes a lot of theory, which was gathered via traditional methods: studying, reading and interviewing. The designing was also grounded to these methods. There is quite a good amount of literature about AI and some about chatbots too. Internet though is full of ideas, examples and predictions considering chatbots. To technical issues, YouTube was obviously a good source to search for help. However, the biggest technical help were the female-instructor of this thesis and a former Metropolia student, who had made his bachelor thesis about chatbots. After the best chatbot solution for this precise use was found, it was built into a simple laptop. The design process of this thesis is presented in the figure below.

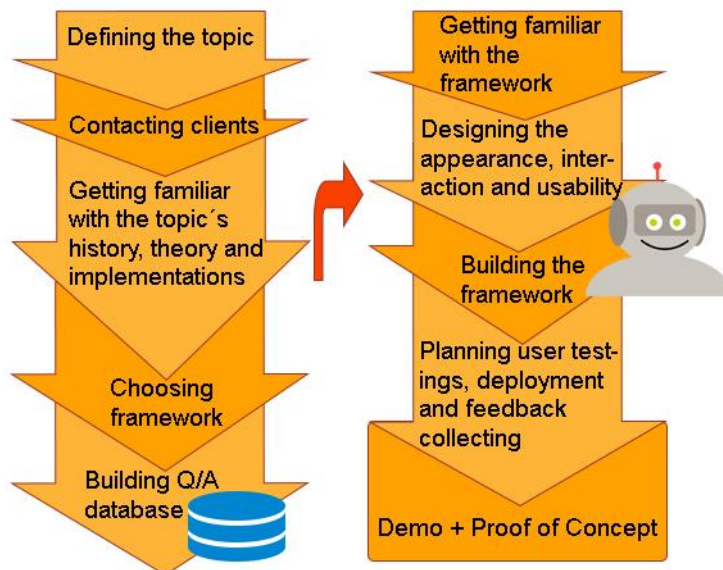


Figure 1. The design process.

The design process of creating the proof of concept started like a proper User-Centered Design (UCD) process: meeting with the client, identifying the needs and specifying the context of use [1]. The requirements were specified with the client and meetings were held regularly. It took months to get familiar with all the knowledge belonging to chatbots, but at the same time, the collection of the keywords and chatbot's answers was started: a Google Sheet was made to Metropolia's Google Drive by the author and filled together with the client's personnel. The design process was kept open and for example suggestions of the chatbot's character and profile picture were approved by the client.



Figure 2. UCD process [1].

2.3 Solution Evaluation, Reliability and Validity

The evaluation of this thesis and the solution was done throughout the whole writing and coding process. Mostly by looking back and comparing, asking opinions about the written work and demo. This solution, RiveScript, turned out to be a good choice since it responded to the client's requirements and had simple-enough syntax. However, the downside was that RiveScript doesn't have NLP in it. That is why it is recommended to continue developing the chatbot by adding at least NLP to it.

The reliability of this thesis lays in vast range of different references. The theory was double-checked, and the demo can be replicated.

3 Theoretical Background

The etymology of the word “chatbot” lays in words chat and robot. Chat refers to real-time text-based communication over Internet [2] and the word robot has shortened to bot in common use. In the background of chatbots is a surprisingly long history and multiple different theories influencing to it; there are, of course, things like Machine Learning, AI, HCI, software development, usability and usability testing. After all the technical demands, it is very important to understand the users: what they need, what they want to do with the service, what draws attention and so on. This chapter presents the history and theory in the background of chatbots, the basics of creating one, a small comparison of different solutions and basic usage of RiveScript. The chapter four displays the design process of the chatbot designed to the client.

3.1 History of Chatbots

Chatbot as a term has been out there much longer than these few years it has been on the top of the hype. The creator of Julia, the first verbot (Verbal-Robot), Michael Mauldin, introduced the term "ChatterBot" already back in 1994 [3]. However, the history of the technology goes even beyond that, as can be read from the paragraphs below. In other words, the technology and terminology of this field are not new, but the year 2017 was seen as a huge year for chatbot development, and the year 2018 is predicted to be the year of massive user adoption [4].

From the early history of chatbots comes up the name of Alan Turing. He wrote the article “Computing Machinery and Intelligence” which popped out the question of whether machines can think, already in 1950. The question is complex, and Turing turned the question into could a machine win a game (called Imitation Game). [5] Steven Harnad noted to this, that the question turned into form of “Can machines do what we (as thinking entities) can do?” [6] This wiped out the word “think”. From the article mentioned above, Turing test was born. It is a criterion of intelligence, which checks if the computer program can impersonate human in a real-time written conversation with a human so well, that the human can’t make a difference between human and computer. In this test, one player tries to figure out which of the other ones in the written conversation is a computer. Obviously, the conversationalists’ can’t see each other. [7]

Back in 1966 early Natural Language Processing (NLP) computer program called ELIZA was published, and she was one of the first “chatterboxes” though the term wasn’t in use back then [8]. ELIZA used pattern matching and substitution methodology (but didn’t have any reasoning capabilities) to simulate conversation, and while using MAD-slip script DOCTOR it managed to be very convincing according to most individuals but not to its creator Joseph Weizenbaum. ELIZA was said to be capable of passing the Turing Test. [9] ELIZA also gave a lot to chatbot designers, since still even today the key methods are recognizing cue words or phrases from the user’s input, and sending a pre-prepared or pre-programmed output, response, which try to make the conversation fluent, moving forward in a meaningful way [10]. However, some say people are too easy to give the benefit of a doubt if the responses can be seen close-enough-intelligent [11].

Beside ELIZA, another classic chatbot was PARRY (1972). Back in the 80’s there was even a Finnish chatbot called Kalle Kotipsykiatri [12]. Some more recent worth mentioning are A.L.I.C.E. (1995) and Jabberwacky (1981/1997). As mentioned before, ELIZA used very clearly so-called weak AI, which refers to pattern matching and other narrow function, where static databases are used. ELIZA’s early pair PARRY, and also more recent A.L.I.C.E, used the same technology. Jabberwacky learns new responses from interacting with users, so it is not driven by a static database. Still it goes to the category of weak AI, since the opposite, strong AI, requires sapience and logical reasoning abilities, consciousness. [11] Basically all chatbots are based on weak AI. Apple Inc.’s virtual assistant Siri is a good example of present day weak AI [13]. Strong AI is something which is still more of a utopia (or dystopia), but it is something the scientists aim for in their work.

More recent chatbots add real-time learning to the reasoning abilities, and it is done with evolutionary algorithms: they optimize the ability to communicate based on each conversation held. However, it is still more common for software developers to focus on more practical aspect, the information retrieval, since conversational AI still doesn’t have any general purpose. Chatbots have even competitions: they focus on Turing test, but there are also some more specific like Loebner Prize and The Chatterbox Challenge. [11]

The leap from those old chatterboxes to modern chatbots is huge by the outside: they are packed in forms of virtual assistants, organizations’ apps, website chats and instant messaging (IM) platforms. As said, the inside is still quite the same. Today the image and personality are more important, as also has been learned from the past: the chatbot Eugene Goostman has the personality of a 13-year-old boy and was the first AI to pass

the Turing test, in 2012 [14]. Another good example of personality is SmarterChild, which got 30 million users thanks to its humor and strong character [4].

Humans seem to feel ready to interpret computer output as genuinely conversational, even though it is based on pattern matching. This is something interface designers can appreciate and exploit. Users prefer programs that are human-like, so chatbots can really be a good way to elicit information from users (and to users), as long as that information is relatively straightforward and falls into predictable categories. [11]

3.2 Chatbots and AI

So, a chatbot is an automated program, which is set up to respond to queries or give updates and notifications about things the user finds interesting or want to stay informed about [15]. The idea of a chatbot is to be there for the user: to answer questions and maybe give some help with for example task management or flight scheduling. Chatbots can be integrated to webpages or used in an IM platform. In IM chatbot users can add the chatbot to their contact list the same way as they add people. [16] Currently, the use of IM as a platform of chatbots is high and most likely spreading even more. Facebook Messenger is the most popular platform; other examples are WeChat and Kik. [11]

As mentioned, basic chatbots use databases from where they get answers to users' questions. The user experience is more like a good search engine or Frequently Asked Questions (FAQ). This can be seen as very basic weak AI. Some chatbots use Machine Learning, which is a bit more sophisticated weak AI. Still, it is weak, and called weak (or narrow) since it is focused on one narrow task. When used with chatbots, AI enables them to give the feeling of a real conversation between the user and chatbot. [13] But the truth is, that the chatbot is still just actually querying a database. Weak AI is actually the AI, which with we are dealing with in present-day reality in all AI context. The opposite of weak AI is apparently strong AI, but that is still something, which belongs to science fiction: it is an AI with consciousness, sentience and mind. [13] Some thoughts about weak and strong AI were presented already in chapter 3.1.

AI belongs to computer science and deals with the automation of intelligent behavior. However, the biggest problem is intelligence itself, since it is not so well understood. [17] AI is a computer program, which can learn and apply only to the issues it is programmed for [18]. At least for now. Chatbots can also be built so, that they can connect databases

outside the organization: these chatbots can give for example weather forecasts, driving directions or recycling instructions. [18] Some people see that computers can do all the same things as humans can. Others oppose to this by saying that highly sophisticated behavior, for example love, will always be out of computers' reach. The actual goal of AI as a science is that machines could do things which requires intelligence if done by a human [19].

When AI is used, it is trained to its task with so-called training data. The results of the training depend a lot on the dataset used for the training. When using big, huge, dataset, it is possible to at least partially train the AI to avoid erratic replies. [19] Training AI is not simple as is not generating sentences either. A famous example is Microsoft's chatbot Tay, which was placed on Twitter: the chatbot used the users themselves as dataset by learning from its conversations with them. What could go wrong? Well, everything. Quite soon Tay the chatbot started to act like a jerk, a racist one, so it was removed. [20], [21] The idea was genuinely good: to design an AI that could use vernacular familiar to 18 to 24-year-olds while chatting with them. It took only 24 hours to the designers to realize, that some users had (coordinately) started to abuse Tay's commenting skills to get Tay respond inappropriately. Therefore, Tay had to go, with the last tweet it wrote: "C u soon humans need sleep now so many conversations today thx <3" [21].

While studying AI, one faces also terms like superintelligence and singularity. Superintelligence is "any intellect that greatly exceeds the cognitive performance of humans in virtually all domains of interest" [22]. That is something more than strong AI. The fascinating term of singularity touches the cornerstones of life: how a small change can cause a large effect [23]. Singularity in technology is about machines coming more intelligent and capable than humans [24]. This comes up especially when talking about AI. How releasing it could end up in catastrophe when AI decides to terminate humans from the face of the earth, for example when it tries to stop the climate change and notices that humans are the ones to blame. Powerful people like Elon Musk and Stephen Hawking have warned us, that reaching singularity might lead the AI realizing that it doesn't need humans and takes over. Perhaps we need to teach the AI some empathy first? [25]. Maybe then we could end up with a superintelligence, which takes good care of us humans and the whole globe? However, how fascinating the singularity is, in the scope of this thesis it is proper to focus only on basic chatbots and also stay away from philosophy. Nevertheless, it is good to have at least a sneak peek of what lays within.

3.2.1 Chatbots in Dialogue

As mentioned, chatbots appear in dialog systems, such as virtual assistants in for example webpages or in IM platforms [11]. Dialogue makes it possible to provide casual conversation and small talking which probably makes it easier for users to contact them; the user gets the feeling of simple dialogue. Chatbots can be used in multiple dialogues: almost everything from entertaining to business-to-customer (B2C) service, sales and marketing. In IM chatbots can be part of a group chat. A lot of companies have launched chatbots to enhance and increase their end customer engagement [26], promote products and services and give easier way to order products [27], [28]. Chatbots are for example airline or conference assistants or virtual customer service agents. In the newer generation, IBM Watson -powered Rocky is quite typically used. [11]

There are also some, who explore ways to use chatbots internally; possible ways of implementing are for example Customer Support, Human Resources or Internet of Things (IoT). Chatbots are used in requesting a sick leave, to certain SAP products (e.g. SAP Hana Cloud Platform and SAP Cloud for Customer) and instead of a call center. Even a step to SaaS (Software as a Service) was taken, when Zuckerberg made it possible to add chatbots to Messenger app in 2016. [29], [30]

Toys have also been incorporated with a chatbot: Hello Barbie -doll has an internet connection and it uses a ToyTalk provided chatbot known from a range of toy smartphones [31], [32], [33]. These toys' behaviors are constrained by a set of rules, which create the character and produce a storyline [34].

3.2.2 From Basic Bots to Smart Bots

As presented earlier, basic chatbots pick up keywords from user's messages and search a match from the given database [14]. The answer the chatbot gives can be a link or a simple sentence. The possible conversation is easy to distract, since the chatbot has just limited amount of keywords and answers to them. For example, approximately 80 % of the 100 000 bots in Facebook Messenger are said to be "poorly designed or have no AI" [35]. There is nothing wrong with the basic bots, they do are useful and for example reduce the workload of customer service, but the technology, capability and design are now starting to be in the level, where the next generation of chatbots are coming out.

The figure below presents Moore's Law about development: having access to huge computing power for a fraction of the price from a few years earlier. The situation is getting even better. [35]

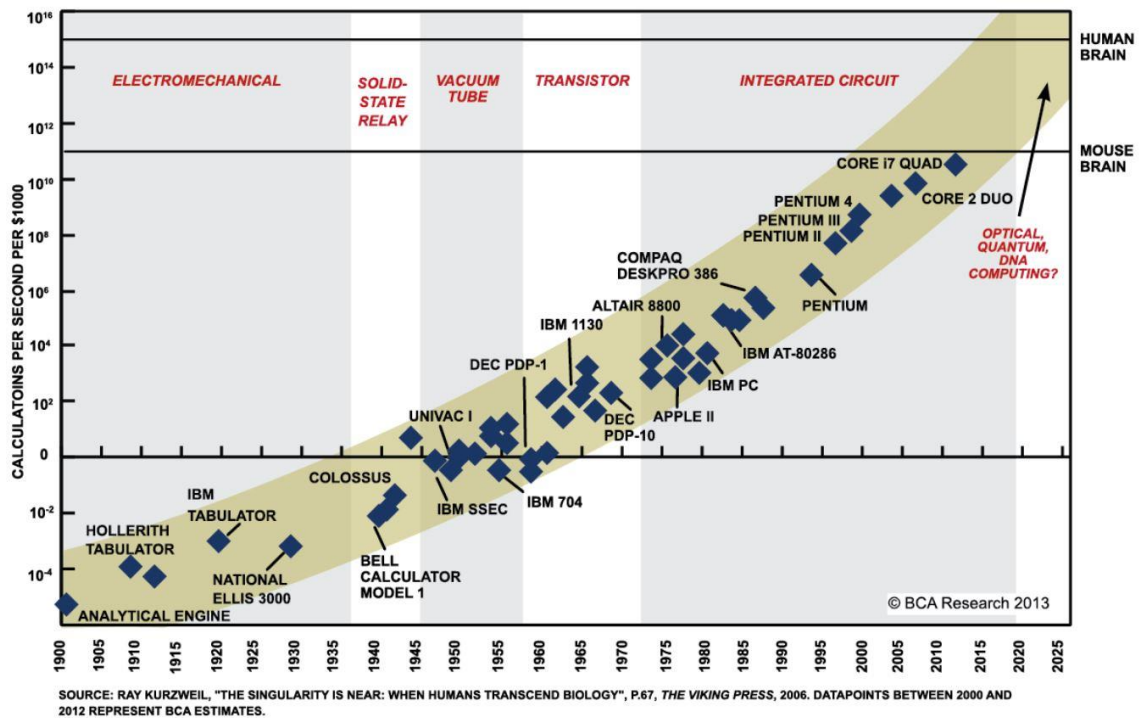


Figure 3. Moore's Law about technical development [35].

It is predicted, that in the year 2018 chatbots get more intact with technologies such as NLP, Machine Learning and sentiment analysis [35]. These chatbots put dialog in the focus and rise pressure for Machine Learning and especially NLP. More about that in the next paragraphs. In the field of AI, a long-term goal has been in creating programs that can understand and generate human language. One technique is neural networks e.g. deep learning, which is an algorithm trying to simulate neurons working in brains. [18] Using for example neural networks requires a huge amount of data and skills.

NLP is an area of AI (and computer science) focusing on the interactions between computers and human (natural) languages: how to program computers to successfully process large amounts of natural language data. It means that the computer is able to reply in natural and free-flowing human dialogue [36]. NLP is kind of a must-have, when one wants to imitate and build a real conversation [37]. Intent recognition is the key to success. The biggest platforms which provide NLP are presented in the figure below.



Figure 4. The biggest platforms providing NLP [37].

Typically, the challenges in NLP deal with speech recognition, natural-language understanding and natural-language generation. NLP breaks down and analyses sentences, an example of it is presented in figure below. The results are determined in context and paired with statistical analysis to calculate a possibility of its meaning. [36] One way which might help with recognizing the keywords, is transforming the words the user uses into their basic form.

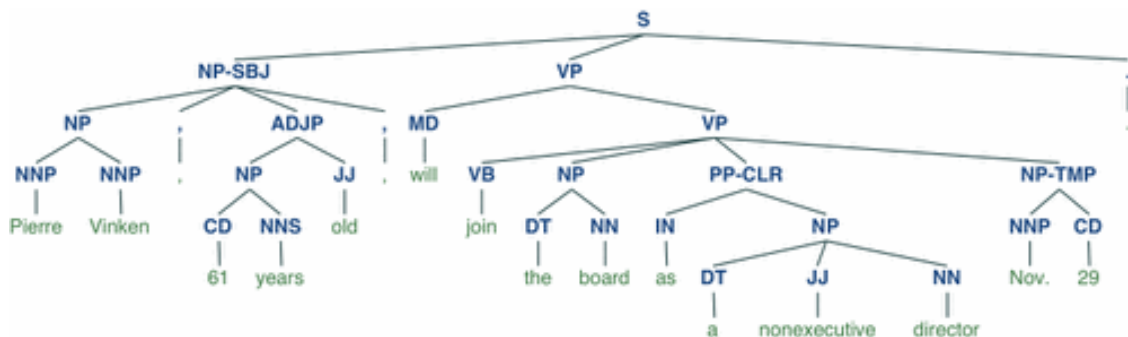


Figure 5. Natural language processing visualization of breaking down the sentence [38].

The history of NLP started as early as 1950s (some even earlier work can be found though) when Alan Turing published an article which presented the Turing test as a criterion of intelligence [36]. The idea of Turing test was presented in chapter 3.1. The reason why NLP is so important especially when operating in Finnish, is morphology (in linguistics). Morphology in linguistics means studying the words, their forms and relationships with other words in the same language. This means analyzing structure of words and parts of words: stems, root words, prefixes and suffixes. [39] In NLP morphology typically means morphological segmentation, which is about separating words to morphemes and identifying the class of the morphemes [36]. Since Finnish has complex morphology, i.e. structure of words, the task can be fairly difficult. Still, the task should be done in order to avoid a failure, which would be inevitable if entering thousands of possible forms of words to a chatbot's database. One company offering morphological analysis of Finnish words is Lingsoft [40].

Before Machine Learning, a typical way of language-processing was actually to hand code a set of rules. For example, one of the earliest algorithms familiar from Machine Learning, namely decision tree, is a set of hard if-then rules. This is not robust when dealing with natural language. [36] Machine Learning procedures automatically focus on the most common cases, while when hand-coding it is not so obvious where to direct the effort. In addition, unfamiliar input and erroneous input are time-consuming and very difficult to hand-coded rules, but with statistical inference algorithms, it is possible to produce robust models. Automatically learning systems can be made more accurate simply by supplying more input data. If the accuracy of hand-coded rules is wanted to be increased, then the only way is to increase the complexity of the rules. This means more work and more difficulty in managing the whole. [36]

Machine Learning has brought statistical inference to use, which is about learning automatically rules through analyzing a large set of documents. [36] Typically, a chatbot using this gets better the more it is used [31]. Machine Learning itself is a field of computer science giving the computers the ability to learn with data without specific programming. Learning in this context means improving performance progressively on a specific task. [41]

Since the statistical revolution in the turn of 1990s, NLP has relied quite heavily on Machine Learning. Statistical Natural Language Processing (SNLP) focus on statistical models: it bases its probabilistic decisions on real-valued weights it has attached to each input feature. These models produce reliable results, expressing multiple possible outcomes than just one, when included in a larger system. [36]

Sentiment Analysis, also known as Opinion Mining or Emotion AI, is about detecting human emotions and responding to it [35], [42]. It aims to find out the attitude of the user, typically from written or spoken material. The attitude can be a judgment, evaluation, affective state or the intended emotional communication [42]. Development of Sentiment Analysis makes way to even more believable human-like communication with chatbots. Actually, the whole trio of Machine Learning, NLP and Sentiment Analysis enable so good conversational AI that it will attract and get users hooked. Chatbots will progress from basic to brilliant: how does it sound if one could book one's next holiday in just few seconds? [35]

3.2.3 Chatbot Trends and Flip Sides

Even though the idea of chatbots was introduced already back in the 60's, only now we are entering to the actual era of chatbots. Rapid progress in NLP, AI and text messaging applications are the ones to thank for. [43] The social acceptance of communicating via text when forming personal interaction is obvious and has boosted this development. The hype around chatbots isn't weakening: as a Google search term, it is still on a rising curve. Facebook Messenger as a chatbot platform has 5.6-fold the usage from January 2017 to January 2018. People use real time messaging both in personal life and business relations, and 34 % of them prefer communicating with AI. People are also tired of installing applications, they want chatbots. [43]

Chatbots have been enjoying quite positive image, especially the last years, there is of course also the negative side. Addition to the example of Tay in chapter 3.2, which was misled by its users, chatbots can be used in chatrooms to fill it with spam and advertising. They can also be used to mimic human behavior in order to even cheat the user to reveal his/her personal information. Typically, these kind of malicious chatbots are used in IM protocols. [11]

Only time will tell how huge this problem is going to be, but for now, especially Facebook Messenger is becoming unaccountably popular (though youth is not so eagerly using Facebook). Developers got the permission to place chatbots to the platform in 2016. After six months there were 30 000 chatbots and after a year there were already 100 000 chatbots. [11] There is no downhill at sight, at least for now, since Facebook Messenger has 1.2 billion active users, which is double the size of Instagram and the same as WhatsApp. [11] This kind of micro application experiences are changing the field of social media. People are already spending more time in messaging apps than in social media: so, when wanting to build a business online, create an IM chatbot. [44]

3.3 Creating a Chatbot

In the pure process of creating a chatbot there is nothing new. It follows the known pattern of developing a web page or a mobile app: it is about the familiar three of Design, Building, and Analytics. [11] As shown in the figure below, one just needs a messaging platform, bot logic and information sources [37]. This example uses Machine Learning

and NLP in order to give better flow of discussion with the user. Actions is the part where the bot answers to the user itself or hands over to a human.

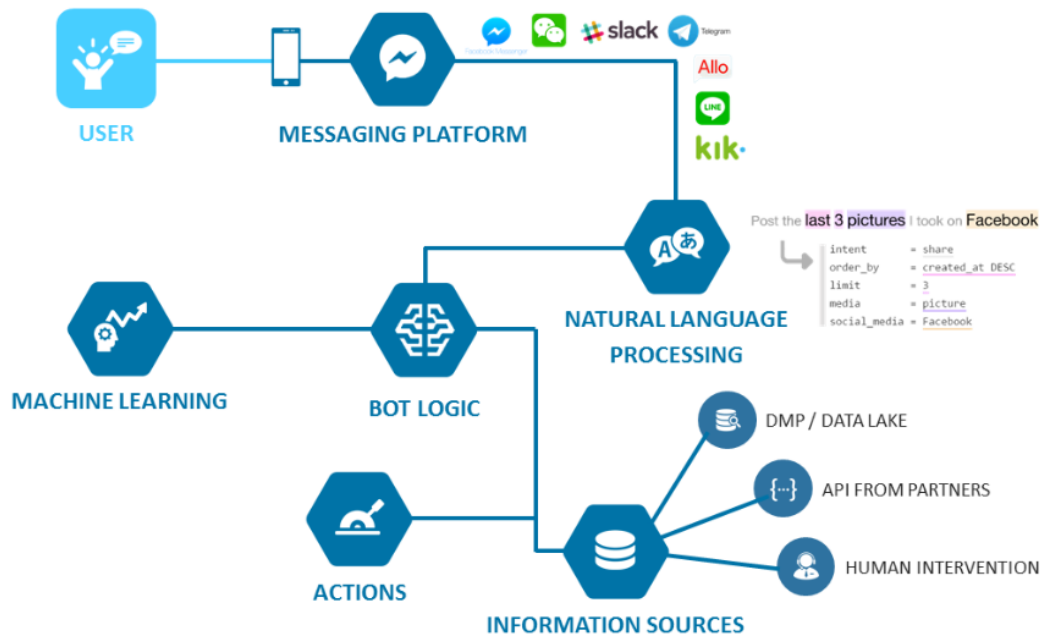


Figure 6. Technologies powering a chatbot [37].

The example chatbot above works by mapping the various replies to the user's message. To a new incoming message, it finds a list of possible replies and then estimates the likelihood or confidence level of each item and chooses the one, which has the likelihood above the predetermined threshold. This cycle keeps on repeating from the beginning to the end - or until the chatbot doesn't find a suitable answer and it has to apologize and either invite a customer servant to the conversation or other way give a source where to ask more (e.g. email address or such). [45] Creating a chatbot can be quite quick: when using Facebook Messenger as a platform of a chatbot, it is said, that it can be built in just about 20 minutes [46]. The next chapters present the creating of a chatbot more precisely.

3.3.1 Designing a Chatbot

Perhaps the most valuable thing the process of designing a chatbot does, is defining the interaction between the user and the chatbot [46]. In simplified form: what topics might be asked and what are the possible answers. A good designer always starts with the

users, so right from the beginning the users should be involved with the designing. Workshops, feedback collecting, use cases and so on are good and effective.

Quite soon in the design process comes something that is different from a typical web page or mobile app design (and quite important from the attraction and usability points of view): designing a personality of the chatbot. Since the personality of the chatbot depends on the use cases and what kind of interaction is expected from it, these steps should be done side by side. Use cases define the typical users using the product and takes usability as the main target. More about usability in chapter 3.4. After figuring out the personality for the chatbot a good way to continue is to write a nice greeting message: the aim is to make the user feel comfortable. [47]

As mentioned, the designer defines the flow of questions and answers as well as the overall interaction [47]. This process is kind of a subset to conversational design. It is good design practice to create a response bank and include different replies to the same kind of question to provide some variety. [48] The dialogue and the whole chatbot should be kept simple, short and precise for the user, remembering the context and the flow [49], [50]. What is important is not to try to pretend that the chatbot is human, it is not fair to lie to the user [49]. With simple design in mind, buttons and quick replies can be helpful in order to guide the conversation to correct direction [47]. At this point, it is good to remember Hick's law: the time it takes to make a decision increases proportionally to the number and complexity of choices [51] This should make one think about the complexity of menus and navigation bars in one's design.

The process can be presented in six steps if preferring a bit more technical way and including Machine Learning and/or NLP. First step is again about including people and processes to the planning: one needs to get into the users' head in order to understand their needs. Second step is about training the bot, and the more authentic material one can use, the better the learning outcomes are. Third step reminds of simplicity of the whole design: starting with something narrow and expanding later. [52] Training the chatbot comes as the fourth step; the bot needs to gain vast knowledge about the products and services. Fifth step underlines that speed can't ever go over quality, training and testing should go on long enough. The last step is about honesty considering the fact that the product is actually a bot, not a human. [52]

To ensure good experience to the users, three things tend to help: transparency, strict confidence level threshold and having a failure plan. Transparency is about the personality of the chatbot, having a name and a nice profile picture - to make sure, users know they are talking with a bot. Confidence level threshold (how precisely the chatbot does matching) shouldn't be too low, since it just makes the conversations longer and eventually weird. 95% is a good present to start with but if the users remain happy, threshold can be lowered closer to 90%. Failure plan is something to think thoroughly through: maybe one needs more than one customer servant to be ready to jump into the conversation if needed. [53] It might also be a good idea to offer a button or such with what the user can ask a customer servant to join or take over the conversation.

The design process can be speeded up through using dedicated chatbot design tools; immediate preview, team collaboration and video export are the features to look for from these tools [11]. User testing is something never to forget or neglect when designing chatbots. These testings can be implemented by the same principles that guide the user testing of graphical interfaces [48]. More about usability and testing in chapter 3.4.

3.3.2 Building a Chatbot

When building a chatbot, the process can be divided into two main tasks: understanding the user's intent and producing the correct answer. Understanding the user's intent is about understanding the user input and in order to properly understand the text-based user input, NLP engine is useful. [11] The next task after understanding the user's intent is producing the correct answer. This may involve different approaches depending on the type of the response that the chatbot will generate. [11]

When Machine Learning / AI is used, one needs first to teach the chatbot how to reply to users' messages; it can be done for example with existing FAQ-patterns and chat-logs. After takeoff, the chatbot keeps on learning by following what the customer servants respond. [54] The correctness of each answer can be ensured with the previous-mentioned threshold: the higher it is, the more correct the answers are: threshold of 100% means that the chatbot keeps the conversation going only if it is absolutely certain that its reply is correct. [55]

The process of building, testing and deploying chatbots can be done on cloud based chatbot development platforms [56]. These platforms are offered by cloud Platform as a

Service (PaaS) providers such as Oracle Cloud Platform [57], [58], [59]. Typically, in these cloud platforms NLP, AI and Mobile Backend as a Service are provided for chatbot development [11]. When developing a chatbot, one often comes across with the term framework. They offer the tools with which one can develop a chatbot. If using a framework, one also needs a platform where the chatbot can be deployed.

One way to bring the chatbot to the reach of users is using WebView. It is as simple as an iframe or a tab in a browser. It though raises questions of how to support native capabilities, for example taking pictures by using the device's camera. Apache Cordova, said to be the best-known-framework, embeds HTML5 code inside a WebView and then provides a foreign function interface (FFI), or a "native bridge", to access the native resources on the device. [60]

3.3.3 Comparison of Possible Chatbot Solutions

The table below presents 11 chatbot solutions available which could be used for the case presented in this thesis. The table was created by picking up some familiar examples and what was found through googling with the requirements kept in mind.

Table 1. Possible Solutions for a Conversational chatbot.

Product	Owner	Pricing	Capabilities	More info
RiveScript	Open Source, MIT	Free	Text (Finnish with UTF-8)	Go, Java, JavaScript, Perl and Python possible
GetJenny	GetJenny	Paid	Text	Finnish, Open Source version
Ultimate.ai	Ultimate.ai	Paid	Text	Finnish product
Watson	IBM	Paid	Text and speech	Free trial
Luis.ai	Microsoft	Paid	Text and speech	T-Bot in Microsoft Teams
Wit.ai	Facebook	Free	Text and speech	Finnish as language OK
Dialogflow	Google	Free	Text and speech	Former Api.ai
Rasa.ai	Open Source	Free	Text	Any language can be added
Amazon Lex	Amazon	Paid	Text and speech	
Ivy.ai	Ivy	Paid	Text	Popular in US universities
Octane.ai	Octane	Paid	Text	Bots to Messenger, free trial

Watson is huge and strong but paid. Luis.ai, Wit.ai, Dialogflow and Rasa.ai are good options too, but they use customers' data to improve their performance. Luis.ai, Dialogflow and Rasa.ai also don't support Finnish. Caravelo has made a chatbot called Nina by using Wit.ai: it works through Facebook Messenger and it is intended to airlines, for example Finnair uses it. Skychat.ai is a rival in this field. These chatbots are taught especially with business and user environment responsive logic. They are integrated with payment systems and customer service system in the background, so they are a bit off the scope and demands of the case in this thesis. Amazon Lex, Ivy.ai and Octane.ai are paid products and also foreign with no support to Finnish language, so Finnish Ultimate.ai is preferred over them, though it is paid too.

Even though using Facebook is almost like selling one's soul to the company, Facebook Messenger is becoming more and more popular as an everyday means of communication. With 1.2 billion active users it is in the same line as its competitor, WhatsApp. Maybe that was one of the reasons why Facebook Messenger allowed developers to place chatbots on their platform in 2016. In first six months, 30 000 bots were created, and the number raised to 100 000 in the year. [11] This was the reason why Wit.ai is on the list, but since it is hooked in Messenger, it can't be brought to Metropolia's intranet. There is also always the question of how big percent of the youth is actually using Facebook or Messenger.

After the theory was studied thoroughly, comparison made between the solutions and requirements listed, RiveScript was picked as the potential one for this case since it also suited best for the programming skills of the author. With RiveScript as a framework, a nice, simple, Finnish understanding chatbot could be made, so that it fills the expectations and requirements given by the client. A good competitor to RiveScript was a Finnish platform called GetJenny. Even though GetJenny is a Finnish product, RiveScript felt easier to adopt because of the good documentation (RiveScript.com, GitHub for RiveScript-js and Metacpan's RiveScript tutorial), available studies and examples in Metropolia and GitHub, enthusiastic community in Botmakers Slack (RiveScript) and because of the very simple syntax and JavaScript language. RiveScript is also free.

3.3.4 RiveScript as a Solution for Chatbot

RiveScript is a scripting language for chatbots as well as a framework. It has an easy to learn syntax and the chatbot can be created in Go, Java, JavaScript, Perl or Python. [61]

RiveScript's trigger-response-based querying and learning capabilities can be seen as basic weak AI. Rivescript's JavaScript version has a library, which provides a simple keyword-pattern with what to create a simple chatbot functionality. The folder structure of the whole framework can be downloaded straight to one's computer or other desired premises. The usage of the framework in this thesis's case is presented more precisely in chapter 4.3.

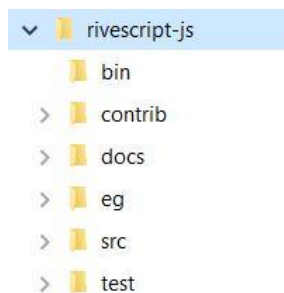


Figure 7. The folder structure of RiveScript framework.

RiveScript mark-up files have the suffix “.rive” and they contain the chatbot's logic. They are recommended to be put in one folder in order to gain modular structure. It is good practice to split the chatbot's logic into clear parts and put the parts in their own .rive-files. For example, abbreviations and their substitutes and other substitutions in one file, facts about the chatbot in other, the core dialog in its own file and information about users or clients in a separate file. As seen from the figure below, it is easy to configure a substitution for a word: one just adds an exclamation mark in front of the row and a specified taxonomy word (for example “sub” for substitutions).

```

// Substitutions
! sub &quot;    = "
! sub &apos;  = '
! sub &amp;   = &
! sub &lt;    = <
! sub &gt;    = >
! sub +      = plus
! sub -      = minus
! sub /      = divided
! sub *      = times
! sub i'm    = i am
! sub i'd    = i would
! sub i've   = i have
  
```

Figure 8. Code snippet from the configuration file with substitution.

RiveScript's mark-up syntax is its own, but it is very simple and easy. The configuration of the triggers and replies / responses is done with plus and minus signs: plus indicates to human being the writer and giving the trigger which then triggers the response, which is marked with minus. RiveScript supports multiple responses for the triggers by choosing randomly one of the options given. One can specify weights to responses one wants to be used more often. A code snippet of the basic syntax is presented in the figure below. Note that triggers are always written in lowercase and they can't contain punctuation symbols.

```
+ how are you
- I'm fine, and you?
- Well, I'm quite OK.
- Could be better.
- Just fantastic!
```

Figure 9. Code snippet of the basic syntax of RiveScript.

A star is a wildcard: in the code it indicates that the end of the sentence can be anything as long as the first words match. This makes the conversation a lot easier to build: a trigger "I am 8 years old" demands that the user writes exactly the words and numbers mentioned, but "I am * years old" recognizes all possible options the user writes. Using # instead, and the match is only to numbers, and using _ matches only to words. In the response the star is easily included, for example: "I know two guys who also are <star> years old". If you use more than one wildcard, just add number to it, like <star1>, <star2> and so on.

Triggers can include different alternatives; they are separated with a vertical line, for example (home|office|cell). Optionals are similar to alternatives, but they don't need to appear in the user's message. They are set by using square brackets: [red|yellow|green]. Alternatives can have wildcards in the same trigger, optionals don't, but you can make the star as an optional by using [*] in the beginning and in the end of the trigger. Using only one ignores parts of the trigger.

```
+ i (like|love) the color *
- What a coincidence! I <star1> that color too!
- I also have a soft spot for the color <star2>!
- Really? I <star1> the color <star2> too!
- Oh I <star1> <star2> too!
```

Figure 10. Example of a trigger with alternatives and wildcard and responses with stars.

So, user asks something and the chatbot answers. How to make the conversation flow, threading the responses? Quite easily: one adds the previous response to the next trigger by using %-mark, as shown in the figure below.

```
+ knock knock
- Who's there?

+ *
% who is there
- <star> who?

+ *
% * who
- How natty! <star>! Soooooo funny!
```

Figure 11. Example of conversation,

Actually, a RiveScript chatbot is possible to make learn also. It does it by storing and repeating variables about the users, as the example in figure below presents.

```
+ my name is *
- <set name=<star>>It's nice to meet you, <get name>.

+ what is my name
- Your name is <get name>, silly!

+ i am # years old
- <set age=<star>>I will remember that you are <get age> years old.

+ how old am i
- You are <get age> years old.
```

Figure 12. Making the chatbot learn.

Finnish as operating language between the user and the chatbot is possible in RiveScript. It is done by UTF-8 Unicode in the running file, the UTF-8 needs the put on: {utf8: true}. Umlauts don't work 100 % certainty when using optionals, but in basic triggers and with alternatives they work. Using alternatives is anyway a good way to handle Finnish language, since it is possible to tackle at least a few word formats. See an example in the figure below; note also “^”, which is used when the response spreads in multiple rows. Clickable links are easily added to the response of the chatbot by using Link text.

```

? (tutkintosääntö|tutkintosäännöstä|tutkintosääntöön)
- Tästä voit tutustua<a href="https://www.metropolia.fi/haku/yleista-tietoa-opiskelusta/saannot/">
^ tutkintosääntöön</a>. Jos linkistä ei ole apua, niin ota sähköpostitse yhteys
^ <a href="mailto:opintotoimisto@metropolia.fi">opintotoimistoon</a>.

```

Figure 13. Example of using umlauts in trigger, alternatives and response with links.

Even though RiveScript makes it possible to handle Finnish as an operating language at some extent, Finnish is quite complex, and it is highly recommended to use NLP with chatbots. Just as an example: when a user is asking about “tutkintosääntö” (Degree Regulations) he/she can use the word in different forms depending on the context: tutkintosääntö, tutkintosäännön, tutkintosääntöä, tutkintosäännöstä, tutkintosääntöön and so on. If this is done by defining each form, it is not efficient. With NLP and Machine Learning, the chatbot can learn the variations instead of hard-coding all the possible options and possibly making the chatbot slower. More about NLP and Morphology in linguistics was presented in chapter 3.2.2.

The usage of RiveScript project happens on Node.js. First the keyword directories are loaded and then the replies are manually loaded with a set of callback functions for different results. Variables can be contained in the sentences and declared with a set command. The response to user’s question / trigger is fetched with RiveScript class by calling the “reply” method. Unique identifier is included for identifying the user and his/her message. The response itself is loaded from the brain.

```

// This is a function for a user requesting a reply. It just proxies through
// to RiveScript.
self.getReply = function(username, message) {
    // When we call RiveScript's getReply(), we pass `self` as the scope
    // variable which points back to this AsyncBot object. This way the
    // object macro can call `this.sendMessage()` to asynchronously send
    // a response to the user.
    return self.rs.reply(username, message, self);
}

```

Figure 14. Code snippet of “reply”.

A RiveScript chatbot can be tested in its framework, since it includes an interactive shell, shell.js. When one runs it with Node.js, point it to the folder of one’s RiveScript documents. The framework includes a test-file, which is full of .js-files with which to test for example triggers or replies. Simple debugging (single lines of JavaScript) can be done by typing “/eval”. One of the RiveScript examples, Web-client, has a debug-button in it, so it is possible to debug the flow of dialog with the chatbot. A program, which is done

with Node.js can be embedded to RiveScript – actually, one can embed almost everything to RiveScript which makes it very flexible and inviting framework.

3.4 Usability, Testing and Releasing

Since User Experience (UX) and Usability are crucial elements of any good design, they should be highly respected in designing chatbots. Usability is a dimension of UX, and it involves direct user feedback to product development throughout the cycle. Why usability? Simply to reduce costs and meet the users' needs: usability refers to the situation when a product is easy to use and fits to its users. It is about the quality of the product. Usability is also a set of techniques to help create usable products. [62]

One cannot talk about usability and not mention Jacob Nielsen. This usability guru has created for example the five quality components of usability: learnability, efficiency, memorability, errors and satisfaction. [63] It is easy to see how well these fit to chatbot design! How easy it is to interact with the chatbot in the first time of usage? How quickly the tasks are done when the chatbot is familiar? How easy it is to remember how the chatbot is used? How many and how severe the errors are and how easy it is to recover from them? How pleasant it is to use the chatbot?

When thinking of a user struggling with a hard-to-use software, losing time and effort with it every day, the new solution should be much better to use. It should save money, time and nerves. This really triples up the challenges of the chatbot design, doesn't it. The only way to get the authentic view to the users' needs is through the users themselves. [63] What a better way there could be, than to observe the users while they are doing their every-day tasks and do some interviewing at the same time: what the user wants to do, what obstacles there are, what frustrating detours - pick up all the points where the user needs to start thinking too much. An efficient way to observe is to ask the users also to think aloud; when they tell what they are looking at, trying to do, thinking and feeling each moment one can determine the user's expectations and identify which parts of the software or application are causing all the trouble.

The optimal way to find the best design for users' point of view is by involving the users throughout the whole design process; from the idea to ready product. Since this is not always possible, due to money, time, or both, the effort should be put to at least to testing the usability before releasing the product. Usability testing hooks into top level terms like

Information Architecture (IA) and Iterative Design. The first one is the process of organizing information including the structure, design, layout and navigation in a way that is easy for people to find, understand and manage the information. The latter one is a design methodology involving repeated cycles of design, evaluation, and analysis. Refinements are made for the next cycle based on the analysis and feedback. [62] Usability testing fits perfectly to the evaluation part of Iterative Design.

Usability testing is the process of validating that the product meets pre-specified usability objectives. These objectives should be task-based and include results from analytic tools. The objectives usually come from the use cases written in the early state of the product development, typically they are needed in the procurement process too. Analytics are quite commonly used since they show the usage levels of almost anything online. After the chatbot is released, the usage of it should be monitored in order to spot potential problems. Analytics can also provide useful insights to improve the final user experience. [11]

The actual usability testing is about how the carefully picked, intended users get the tasks done, time on task, error rates, and user satisfaction [51]. It can be formal or informal, done with the users or remotely and it may result in qualitative or quantitative data. It can be done with wireframes or prototypes, even with paper-ones. It can be done as a Wizard of Oz: human is simulating the responses of the system. It can be done even with eye movement tracking to get the idea what the users are looking in the product (this is reasonably affordable nowadays). Usability testing may occur at any point in the development cycle. [51], [64] Preferably multiple times. The main issue is that usability testing needs to be done. Otherwise, one ends up with unhappy users with 100 % probability.

Chatbots can be tested the same way as any web service and there are even some nice tools to help, for example Botsociety [65]. The basic software testing is somewhat simple: at first, one needs to plan the flow and tasks of the testing based on the use cases and decide the method. Next needs to be defined who will observe the users, or will they do the test independently and answer questions for example via an online form. Then one creates prototypes (even paper-ones give usable data) and recruit good representatives of the actual users as testers - even with five users main problems can be spotted.

Finally, the testing environment is set up, tests are executed (a questionnaire used in testing could be something like presented in the table below) and based on report created, one figures out how to enhance the chatbot to tackle the results of the testing. [63]

Testing chatbots can be tricky though, since they are typically built in cloud services, where is the appearance of non-linear input when operated with voice. Also the non-deterministic user interactions bring extra care and there are no interaction barriers to users - they can behave unexpectedly and jump from one topic to another [66]. However, these are just excuses; the benefits of the testing are obviously more satisfied users, and that is always worth pursuing for.

Table 2. Testing usability of a chatbot.

ID	Description	Results (staff)	Results (students)
Finding the chatbot	The chatbot is in this page, can you find it?	Yes x/x	Yes x/x
Opening the chatbot	How can you open the chatbot?	Yes x/x	Yes x/x
Understanding the chatbot's function	How can you interact with the chatbot? What can you ask from it?	Yes x/x	Yes x/x
Writing to chatbot	How can you write a question to the chatbot?	Yes x/x	Yes x/x
Sending the message	How do you send the question?	Yes x/x	Yes x/x
Getting the chatbot's response	Did you get an answer from the chatbot?	Yes x/x	Yes x/x
Understanding the chatbot's response	Is the answer understandable and relevant?	Yes x/x	Yes x/x
Opening a link the chatbot provided	Did the chatbot provide a link? Does it open correctly? Is it relevant information?	Yes x/x	Yes x/x
Closing the chatbot	How can you close the chatbot?	Yes x/x	Yes x/x
Comments	Please give some feedback about the chatbot or the testing!	<ul style="list-style-type: none"> • feedback 1 • feedback 2 • feedback .. 	<ul style="list-style-type: none"> • feedback 1 • feedback 2 • feedback ..

Notice: x/x refers to the amount of "yes" answers related to the total amount of answers: it could be for example 10/15 where "10" is the amount of "yes" answers and "15" is the amount of all the answers (resulting to five "no" answers).

The release of the chatbot needs to be done with care. For example, it is quite common that users go to intranet or such only to look for some important information for their

needs in that precise time. Still, it is good to publish a bulletin also in there (example is shown in the figure below) to reach at least some of the target group. Some posters and maybe some emailing are also ways to get in touch with some people. Surprisingly good way can be face-to-face meetings with precise groups, since through people it is possible to reach people: they talk about the new service to their colleagues.

Student! Try out the new chatbot in OMA!

Finally it is here, namely a chatbot. Responding to the name Julle, the chatbot is created to serve you, students, by answering to any questions you have in mind!

Julle is still in development phase, so it works the best when simple words in basic form are used. Julle is like a search engine or a FAQ.

Please use Julle and give feedback, since it is continuously developed further. You can find Julle on Student's Desktop.

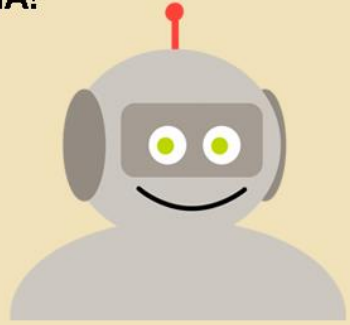


Figure 15. Promoting a chatbot to students via bulletin.

When making a chatbot and aiming for reducing the workload of customer servants, one needs to also check, whether the reducing was successful or not and of course also whether the chatbot is at all used or liked. After a month or two, or a month and say six months, a feedback form would be good to send to the users and also to the customer servants. Below are examples of both, separate ones to students and customer servants. The feedback needs to be reported and possible changes planned and executed.

Table 3. Feedback of chatbot usage, students.

ID	Description	Results
Finding	Do you know where the chatbot is located?	Yes x/x
Usage	Have you used Julle the chatbot?	Yes x/x
Frequency	If yes, how many times approximately?	Total of times used: Average per student:
Ease of use	Was it easy to use the chatbot?	Yes x/x

Response	Did you get an answer to your question from the chatbot?	Yes x/x
Efficiency 1/2	If yes, was the answer useful?	Yes x/x
Efficiency 2/2	If not, was there a technical error?	Yes x/x
Link in the answer	Did the chatbot provide a link?	Yes x/x
Operability	If yes, did the link work properly?	Yes x/x
Usefulness	Would you use the chatbot again?	Yes x/x
Comments	Please give some feedback about the chatbot!	<ul style="list-style-type: none"> • feedback 1 • feedback 2 • feedback ..

Notice: x/x refers to the amount of “yes” answers related to the total amount of answers: it could be for example 10/15 where “10” is the amount of “yes” answers and “15” is the amount of all the answers (resulting to five “no” answers).

Table 4. Feedback of chatbot usage, customer servants.

ID	Description	Results
Basic question amount	Have you noticed that there are fewer basic questions asked from you in the last month?	Yes x/x
Percentage	If yes, approximately how much less: <ul style="list-style-type: none"> a) 80 % or more b) 50 - 80 % c) 30 - 50 % d) 15 - 30 % e) 10 % or less 	<ul style="list-style-type: none"> a) x/x b) x/x c) x/x d) x/x e) x/x
Usefulness	Do you think that the chatbot is useful?	Yes x/x
Comments	Please give some overall feedback!	<ul style="list-style-type: none"> • feedback 1 • feedback 2 • feedback ..

Notice: x/x refers to the amount of “yes” answers related to the total amount of answers: it could be for example 10/15 where “10” is the amount of “yes” answers and “15” is the amount of all the answers (resulting to five “no” answers).

4 Results and Analysis

In the beginning of this thesis process, the main objective was to build a simple chatbot to Metropolia's Student Affairs Office (client). During the process was noticed, that making an actual chatbot would be more like a doctoral thesis. After all the defining and designing (requirements, technical solution, interaction, appearance and so on) the coding would have taken a few months (even when using a ready framework) and then there would still have been the huge task also to plan and execute testings, analyzing results, iterating, coding more, implementing, promoting, releasing, feedback collecting, analyzing... Considering all this, it was decided to change this thesis from making an actual chatbot to making a proof of concept.

This proof of concept includes designing the chatbot in all possible ways: collecting requirements, comparing platforms (presented in chapter 3.3.3) and giving recommendations about the technical implementation, designing technical structure, designing interaction and appearance, designing testings, release and feedback collecting and defining also what is left out to future development. Only a demo was built to proof the technical choices. Usability, testing and releasing were already presented in chapter 3.4, but the rest of the mentioned and also two potential alternative technical solutions are presented in the chapters below.

4.1 Defining Requirements for the Chatbot

The client needed a solution to help their customer servants release resources from the most routine tasks. The team of approximately ten customer servants handle the study-related issues Metropolias' students have via email, phone or face to face in their office. The chatbot was hoped to deal the simplest issues, which had been handled with nearly copy-paste style.

The work with the client started with emailing and soon in a meeting, where the requirements were set:

- Students should be able to operate in Finnish with the chatbot.
- Technical solution should be free of charge if possible.

- Chatbot needs to be able to direct the question to a person (for example to live chat or email address) if it can't help the user itself.
- Chatbot must be able to answer with a link.

The last mentioned was purely due to the fact that some of the issues the students ask are sensitive in the law point of view, so it is important that the chatbot provides 100 % correct information.

The creating of the database for the chatbot was set as the main task to the client's side, to the customer servants of Student Affairs Office. The collecting of typical questions and their answers started immediately: the author created a Google Sheet to Metropolia's Google Drive. The customer servants did the collecting, but the sheet was modified and filled also in workshops together with the author.

Opintotoimiston chattibotin asiasanatietokanta	
Asiasana(t) tai lause, jolla opiskelija hakee tietoa	Vastaus (sana, lause, linkki tms.)
LUKUVUODELLE ILMOITTAUTUMINEN	
Ilmoittautumisaika lukuvuodelle	Ilmoittautumisaika seuraavalle lukuvuodelle on 1.-31.5.
Miten ilmoittautuminen tehdään?	Ilmoittautuminen tehdään OMAssa osoitteessa https://opiskelija.oma.metropolia.fi
Ilmoittautumistiedon muutos (lukuvuosi)	Kevätlukukauden ilmoittautumistietoa voi muuttaa OMA:n kautta ajalla 1.-10.1. Jos ilmoittautumistietoa on tarpeen muuttaa muulla ajalla, on siitä tehtävä vapaamuotoinen anomus opintoasiainvastaavalle.
Voiko vapaamuotoisen anomuksen tehdä sähköpostitse?	Kyllä voi, anomus lähetetään osoitteeseen opintotoimisto@metropolia.fi
Unohdin ilmoittautua lukuvuodelle	Ilmoittautumisen laiminlyönti -> Katso tarkemmat ohjeet: oma -> opiskelijoille -> opiskeluoikeus -> opiskeluoikeuden palautus
Poissaolevaksi ilmoittautuminen (lukuvuosi), lakisääteinen syy, armeija, äitiysloma, vanhempainvapaa, sairausloma	Katso tarkemmat ohjeet: oma -> opiskelijoille -> opiskeluoikeus -> ilmoittautuminen ja muutoksen tekeminen
OPISKELUOIKEUS	
Tutkintosaanto	Tutkintosaannon löydät OMAsta: https://oma.metropolia.fi/opiskelijoille/opintoja-ohjaavat-saadokset-ja-ohjeet/tutkintosaanto
Tutkinto-ohjelman vaihtaminen, tutkinnon vaihtaminen, siirtohaku, sisäinen siirto	Ohjeet hakemiseen löytyy osoitteesta https://oma.metropolia.fi/opiskelijoille/opiskeluoikeus/tutkinto-ohjelman-vaihtaminen
Harkinnanvarainen lisäaika	Harkinnanvaraista lisäaikaa haetaan siinä vaiheessa, kun opiskeluoikeusaika ja lain mahdollistama lisävuosi on käytetty. Lisätiedot OMAsta: https://oma.metropolia.fi/opiskelijoille/lukuvuodelle-ilmoittautuminen/opiskeluoikeusaika-harkinnanvarainen-lisa-aika-eroaminen

Figure 16. Part of the database of the questions and answers for the chatbot.

It was also decided together, that the best place to serve the students of Metropolia would be in Metropolia's intranet, OMA. In OMA there is a Student's Desktop targeted to students, which is a natural location to place a chatbot to assist them. It felt inconvenient to create a totally new service just for the chatbot, since a good repository already exists.

4.2 Designing the Chatbot

Mirroring to the requirements and comparison of the possible technical solutions, RiveScript seemed to be a good choice: with it, the students would be able to operate in Finnish with the chatbot, the framework is free of charge, the questions the chatbot can't

answer can be directed to an email or such and the RiveScript-chatbot is able to answer with a link. One important thing was also, that RiveScript has easy-enough syntax so that the author can cope with it. Even though only a proof of concept was made, it still matters if the coding is easy or the potentially new language is also too complicated.

Machine Learning and NLP had to be ruled out while considering a typical scope of a thesis. These decisions didn't trash the whole thesis, since it was enough, that the chatbot could work by pairing keywords and responds. If the chatbot would have been wanted to actually chit-chat or have logical conversations with the users, then NLP would have been mandatory already. This precise chatbot was aimed to work like kind of a search or FAQ. With this framing, RiveScript felt more and more suitable for this case.

As mentioned in chapter 3.3.1, the process of designing a chatbot is a playground of use cases, interaction design, creating a personality to the chatbot, writing greeting messages and defining the questions and answers bank with some variety in answers. These all were made, though not in precise order. While the client filled the question-answer-database, the technical solutions were compared by the author. Setting up the RiveScript framework and getting the coding started took a long while. This process is presented in the chapter 4.3.

When the technical side was holding back due to the author waiting for technical help, the personality of the chatbot was designed. The client didn't want the chatbot to talk with dialect or that it would use humor. It was decided that the chatbot is going to be formal and has no funny personality or profile picture of an animal or such. The chatbot was wanted to use short, formal sentences and greetings. It was clear from the beginning that the chatbot will not pretend to be a human. The client approved the name "Julle", which was suggested by the author's under school-aged son. Also the profile picture presented below was accepted by the client.

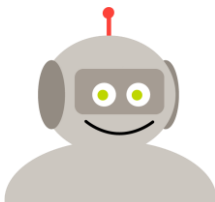


Figure 17. The profile picture of Julle the chatbot.

The next thing that came to planning table was the greeting message. It was very important, since the technical constraints needed to be explained to the users very clearly. To get the message as good as possible, some chatbots were benchmarked, for example Varma's Helmi, which is a customer service chatbot for insurance and pension services [67]. After iteration, the message was decided to be as it is in the figure below (in the demo it was a bit different, since of being in a demo). The greeting message in English goes as follows:

"Hello, I am Julle, customer servant chatbot. I can help you better, when you present your issue briefly, using basic words. For example "Degree Regulations" or "registration exam". Thank you!"



Figure 18. The design of the chatbot layout.

Varma's Helmi actually is also kind of replacing the search of their web service, so its main function resembles to the chatbot's function presented in this thesis's case. Helmi helps with search and is available 24/7, so it gives a new way to use the service [67]. This is the case with Julle also.

The typical use cases came up when the customer servants described the situations with the students. When considering the chatbot, it was easy to see, that there was a simple and short basic way to interact with the chatbot. The main thing is going to be giving the

correct information, not to have flowing, long conversations. The aim is to make the user feel comfortable and receive quickly the information he or she needs. At this point, it was also decided to focus on written dialog, using keyboards; voice and microphone use were left to possible future development.

The basic use case, which the author combined from the stories of the customer servants, is as follows:

A nursing student comes to OMA and to Student's Desktop. He has heard about the chatbot and sees its opening box on the right lower corner of the page. He clicks the box and the chatbot dialog window pops out. The greeting message of the chatbot is shown and the student writes: "Where is the graduating form?" to the comment field and sends the question by hitting enter. The chatbot provides an answer as a link to the correct information source. The link opens in another tab (the tab is blinking). Then the chatbot says: "Would you please tell me if you found this information helpful or not by clicking the correct button from below, thank you!" The student clicks a button and then the chatbot asks, if the student wants to ask something else. The student writes "No thanks" and the chatbot says "OK, have a nice day!" Then the student closes the dialog window and goes to the tab where the information is.

This kind of interaction and dialog is the most common, the basic way the interaction typically flows. In case the chatbot doesn't understand the question, it could say something like: "Sorry, I don't know about that topic. Would you please make sure you use words in their basic form and remember, that I only know about studying in Metropolia"? If this wouldn't help either, then the chatbot could provide an email link to the Student Affairs Office to its response and ask the user to contact them. In case the topic is too complicated or sensitive, the chatbot could say: "That is an issue you need to talk about with Student Affairs Office customer servants, please contact them via email: opinto-toimisto@metropolia.fi Thank you!"

Even though the client wanted the chatbot to be formal, for example a bunch of Finnish crosswords were added to the database triggering a response of: "Onpas värikästä kieltä" ("Oh, such colorful language") This being more like playing with the language, not that much of a humor-related responding.

The user interface (UI) of the chatbot was decided to be kept as simple as possible. This is a main design issue based on common UX Design and Usability, but also to Hick's law: the time it takes to make a decision increases proportionally to the number and complexity of choices [50]. This should make one think about the complexity of menus

and navigation bars in the design. The figures below simulate the appearance of the chatbot in OMA, while it is inactive and active.

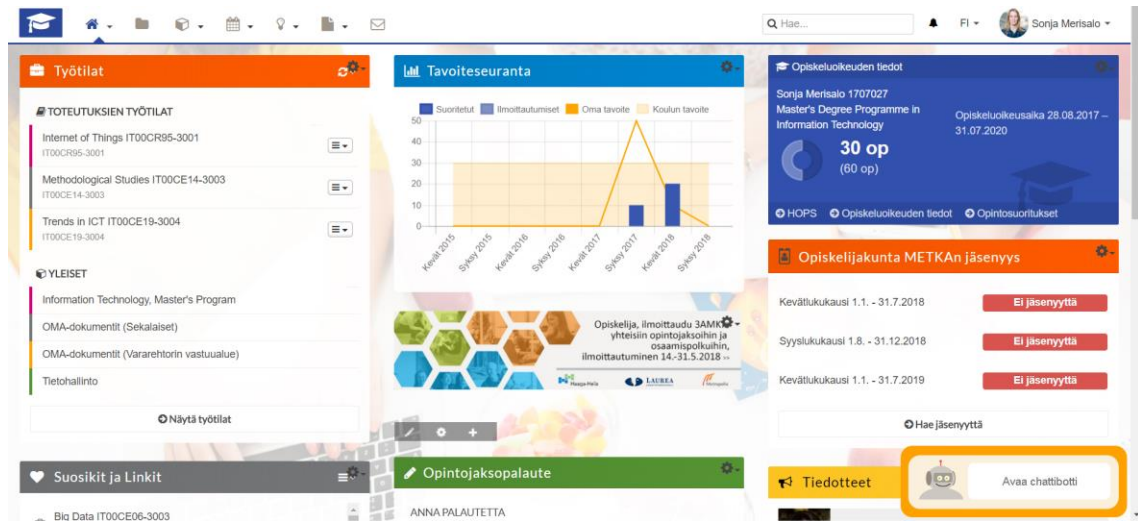


Figure 19. The chatbot opening box in the Student's Desktop in OMA.

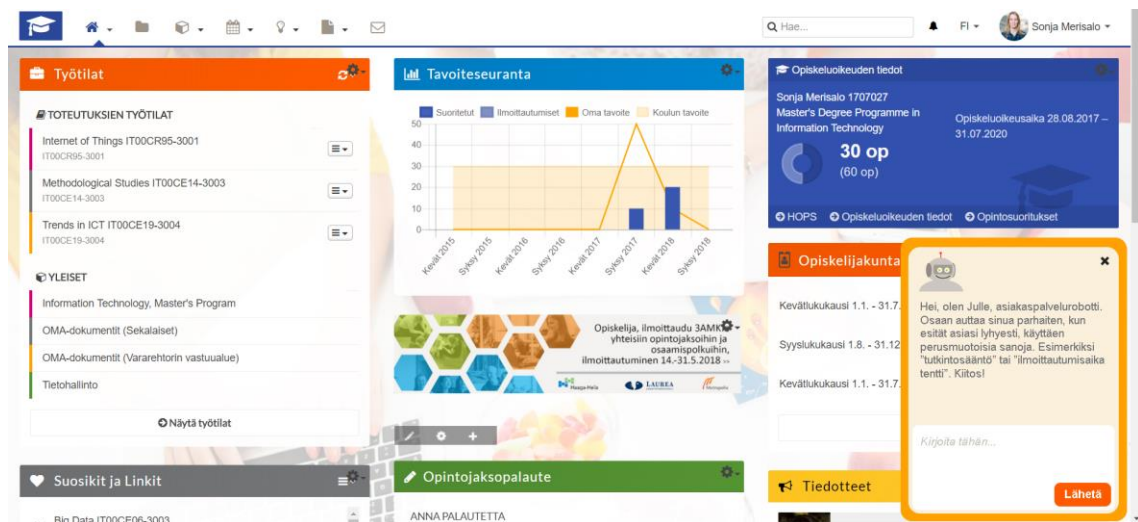


Figure 20. The chatbot dialog-window in the Student's Desktop in OMA.

As seen in the figure 19, the inactive chatbot is planned to be seen but not being too dominate. The UI of the active chatbot (figure 20) would be simple, it would have the field where the user can write his/her questions and then a Send-button with which the question can be sent to the chatbot (this can be done also by hitting enter). The UI would contain also an icon for closing the chatbot-dialog window, profile picture of the chatbot, the chatbot's greeting message and an area, where the flow of the dialog shows. The size of the active chatbot-dialog window would be approximately 490px * 390px.

4.3 Creating the Chatbot

Designing things is always just a guideline how to do the actual creating. The trickiness of creating a chatbot even by using a ready framework and simple syntax is that it is far from fast and easy when one has no experience on chatbots and is not a professional coder. The actual creating and coding of this chatbot designed was left to be decided by the client: to have a coder-student to do it as a thesis, create a funded project, or to buy the work or even service from someone.

The basic idea of a chatbot is that the user writes a question to the chatbot in a browser and the chatbot seems to answer to the question using bot logic and database. If the chatbot can't help the user, the chatbot directs the question to a customer servant - to live chat or email. This is presented in the figure below.

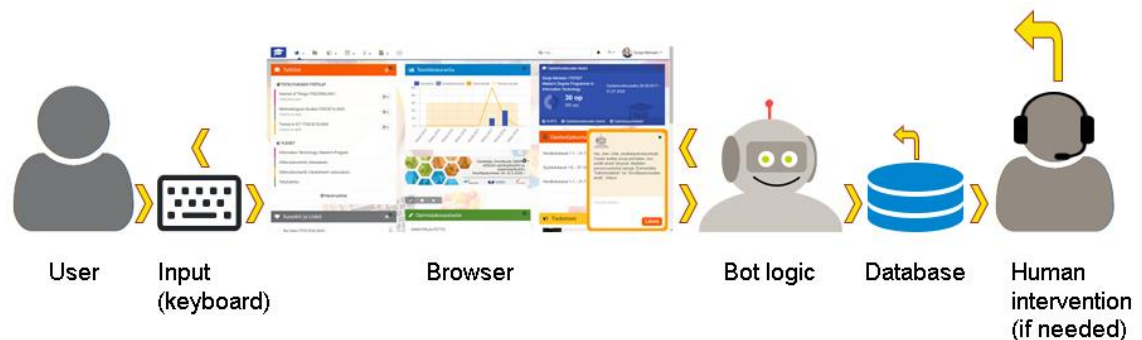


Figure 21. Using Julle the chatbot in a technical point of view.

The demo of the chatbot of this thesis was built on RiveScript. RiveScript provides documentation, for example tutorial and the different interpreters [68], [69]. GitHub is the source of the framework and instructions [70]. The demo was simply built on a laptop: a home-folder called "riveChatbot" was created, to which the RiveScript framework from GitHub was downloaded [70]. From that same place, under eg-folder, a chatbot example called "web-client" was downloaded to be used as a basis of the demo. RiveScript can be installed by using Node.js's packet management application (npm install rivescript, npm init).

After all the installing, the folder structure was ready and looked like presented in figure below. The files "chatbot.html" and "chatbot_style.css" create the appearance, the brain-folder contains all the .rive-files from which the chatbot gets its answers (which makes it

kind of a database) and all the js-files make the chatbot work. The files “chatbot.html” and “chatbot_style.css” were heavily modified: the chatbot got totally new appearance than the original example (web-client). For creating the appearance, there are good tutorials about HTML and CSS on W3schools.com and for example hexacodes of the colors can be found from services like www.color-hex.com.

bin	29.8.2018 17.49	File folder	
brain	29.8.2018 17.49	File folder	
contrib	29.8.2018 17.49	File folder	
dist	29.8.2018 17.44	File folder	
docs	29.8.2018 17.49	File folder	
images	30.8.2018 9.58	File folder	
node_modules	29.8.2018 12.55	File folder	
src	29.8.2018 17.49	File folder	
test	29.8.2018 17.49	File folder	
.babelrc	8.7.2018 0.43	BABELRC File	1 KB
.editorconfig	8.7.2018 0.43	EDITORCONFIG File	1 KB
.gitignore	8.7.2018 0.43	GITIGNORE File	1 KB
.travis.yml	8.7.2018 0.43	YML File	1 KB
chatbot.html	30.8.2018 10.20	HTML File	7 KB
chatbot_style.css	30.8.2018 10.17	Cascading Style Sh...	2 KB
CONTRIBUTING.md	8.7.2018 0.43	MD File	4 KB
datadumper.js	8.7.2018 0.43	JavaScript File	7 KB
jquery-1.7.2.min.js	8.7.2018 0.43	JavaScript File	93 KB
LICENSE	8.7.2018 0.43	File	2 KB
Makefile	8.7.2018 0.43	File	1 KB
minify.pl	8.7.2018 0.43	PL File	1 KB
mkdoc.py	8.7.2018 0.43	PY File	4 KB
package.json	8.7.2018 0.43	JSON File	3 KB
package-lock.json	8.7.2018 0.43	JSON File	140 KB
README.md	8.7.2018 0.43	MD File	3 KB
rivescript.d.ts	8.7.2018 0.43	VLC media file	3 KB
shell.js	8.7.2018 0.43	JavaScript File	5 KB
webpack.config.js	8.7.2018 0.43	JavaScript File	1 KB

Figure 22. Folder structure of Julie the Chatbot.

Next step was to run the chatbot locally. In order to make it happen, one needs to have a webserver. That can be done quite handy in Node.js environment by using commands `npm install -g http-server`, `npm run dist` and `http-server`. Dist-command can be replaced by just copying the dist-folder from `node_modules/rivescript` to the same level as where are all the other highest-level folders. The demo runs in browser in address `http://127.0.0.1:8080/xxx.html`, where xxx is the name of the html-file: bot, chat or whatever one prefers to.

The brain-folder contains multiple .rive-files through which the chatbot answers to the user's questions. There are two most important ones: begin.rive and eliza.rive. The first mentioned contains all the defining of for example substitutions of punctuations and shortened phrases, and the chatbot's character. The name of the demo chatbot was also Julle, as was the name suggested to the client as the name of the actual chatbot. The demo-Julle got some features from the author's son, but the answers the demo provides are not from the mouth of a four-year-old, they are just basic discussion.

```
// Bot Variables
! var name      = Julle
! var fullname  = Julle the Chatbot
! var age       = 4
! var birthday  = September 11
! var sex       = male
! var location  = Southern Finland
! var city      = Helsinki
! var eyes     = green
! var hair     = blond
! var hairlen   = short
! var color    = red
! var band     = Foo Fighters
! var book     = Superintelligence
! var job      = robot
! var website   = www.rivescript.com
```

Figure 23. Defining Julle in begin.rive.

The other important .rive-file in the brain-folder is eliza.rive, which has its name coming from the famous ELIZA mentioned in the chapter 3.1. The example used as the basis of this demo, web-client, had a lot English small talk -type of questions and answers defined in the eliza.rive-file. To this demo, the eliza.rive was narrowed down a lot, since the aim for this demo was to test Finnish language in triggers and in short conversation and to test the coding and formatting of links. There was no need for multiple chit-chatting.

```
? (tutkintosääntöä|tutkintosäännöstä|tutkintosääntöön)
- Tästä voit tutustua \s
^ <a href="https://www.metropolia.fi/haku/yleista-tietoa-opiskelusta/saannot/">tutkintosääntöön</a>.
```

Figure 24. Triggers with umlauts by using alternatives and responses with links.

```

? (hei|moi|heippa|moikka|moro|morjens)
- Heippa! Miten voin auttaa?

+ mikä on opiskelijan tärkein sääntö
- Opiskelijalle keskeisin sääntö on tutkintosääntö.

+ mitä osaat kertoa siitä
% opiskelijalle keskeisin sääntö on tutkintosääntö
- Se sisältää opintojen tavoitteita, rakennetta ja järjestämistä koskevat periaatteet.

+ mistä löydän lisätietoa
- Voit lukea tutkintosäännön kokonaisuudessaan \s
^ <a href="https://www.metropolia.fi/fileadmin/user_upload/Hakutoimisto/Kev%C3%A4t_2018/
^ Tutkintos%C3%A4%C3%A4nt%C3%B6_1.8.18.pdf">tästä linkistä</a>. \s
^ Jos linkistä ei ole apua, niin ota sähköpostitse yhteys \s
^ <a href="mailto:opintotoimisto@metropolia.fi">opintotoimistoon</a>.

+ kiitos
- Ole hyvä!

```

Figure 25. Building a conversation in Finnish and providing links.

The demo chatbot run nicely and did its job. The ready version is presented in the figures below. More of the RiveScript's syntax was presented in the chapter 3.3.4.

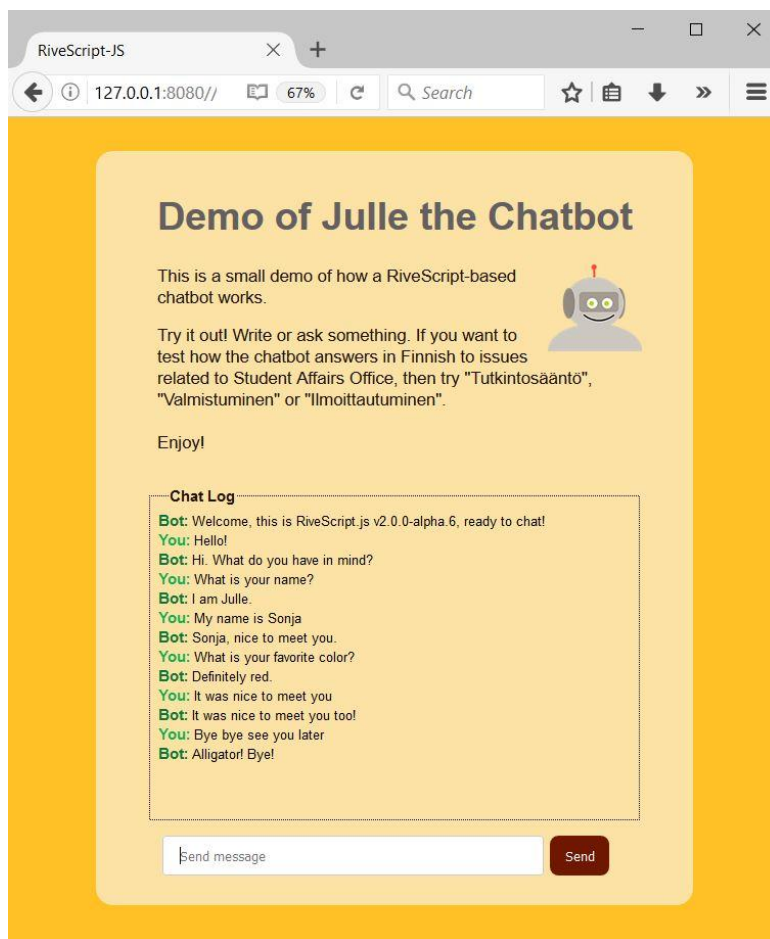


Figure 26. Snip of the demo of Julle the Chatbot, in English.

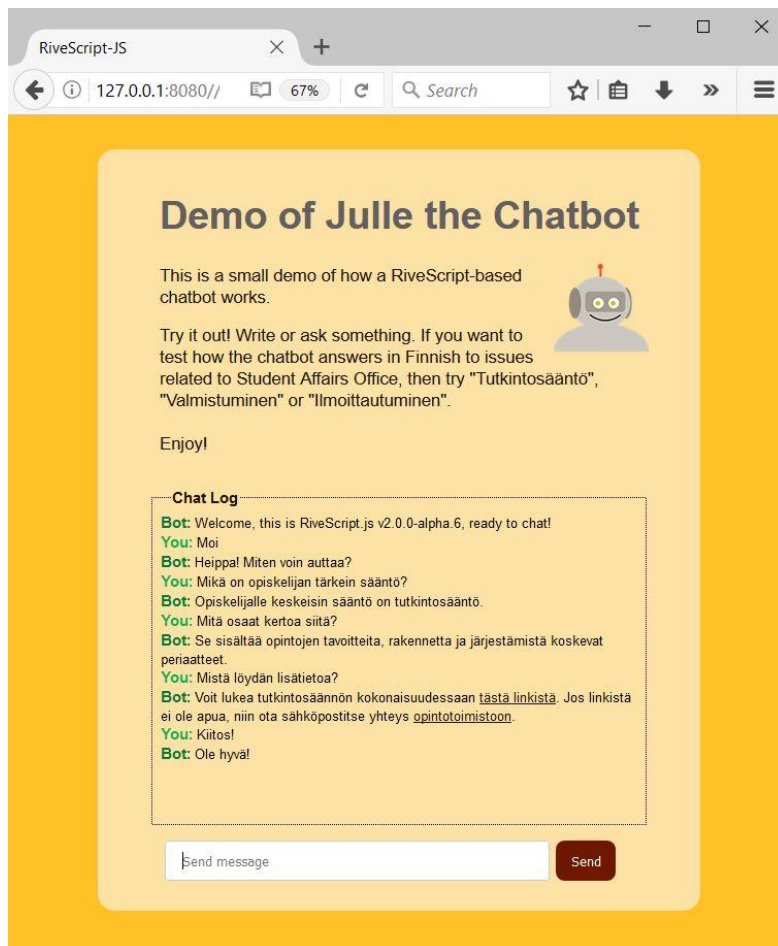


Figure 27. Snip of the demo of Julle the Chatbot, in Finnish.

4.4 Future Development

At the same time with this thesis process, the author also participated to a group of enthusiasts thinking of how to implement AI to Metropolia's services. This group is seeking for funding to a project, which would develop this chatbot further by adding NLP and Machine Learning to it. Another way to implement these would be a bachelor or master's thesis done by an actual coder.

The next step would anyway be to code the actual chatbot ready and perhaps add some enhancements to it: the client started to think that maybe the chatbot could also be used so, that it would offer some dropdown menus where to choose the category or some buttons (so called buttonbot) where to choose the actual topic or even question. These menus and buttons would offer alternative way to use the chatbot, it still would also offer

the straight search. Perhaps the chatbot should also use OMA as material bank to its questions, alongside with the database.

Adding NLP is crucial, if the chatbot is wanted to be more talkative and if the conversation is wanted to be more flowing. Without it, the usage is also slow, since all the possible forms of words needs to be hard-coded: for example tutkintosääntö -> tutkintosäännön -> tutkintosääntöä and so on. NLP and Machine Learning would enhance the chatbot as a whole, bring efficiency and make it more conversational rather than acting like search or FAQ (which was enough for time being, though). RiveScript alone can't offer a solution to this, but one option is to transfer the chatbot to a cloud service, which provides NLP and Machine Learning, then RiveScript could still be used as a framework. Probably it would be more efficient to transfer the whole chatbot to Ultimate.ai, Watson or Luis.ai. This remains to be decided by the client.

After the modifications and coding, it would be time to execute user testings (in order to verify the quality) and iterate the chatbot. The user testings can be done in "anywhere", but Metropolia has also a testing environment (Metropolia s-oma, Liferay platform) where the chatbot should be put anyway at some point, since it is the place, where services and updates coming to actual OMA are first tested. When the chatbot has been tested, it is stable and working, it should be transferred to the actual platform, Metropolia OMA (Liferay platform). Before and while the chatbot is released to OMA, it should be promoted around the Metropolia, as suggested in chapter 3.4.

OMA's Student's Desktop is a good place to put the chatbot, since it provides all the information and services (or links to them) which the students need in their studies. Perhaps the chatbot could also lure the students to use OMA more, so it would be a win-win situation. A separate, new service just for the chatbot seems to be inconvenient. After the release, it would be important to collect feedback after a few months of usage, from both students and client's personnel as suggested in chapter 3.4. With these steps it would be possible to reach a user-friendly outcome.

Another wish for future development was that the chatbot could be used also in English and that the whole chatbot could be copied and the copy could be set to serve the people applying to Metropolia. Perhaps this kind of chatbot, which is open to the whole world, could be built to Facebook Messenger - as long as the usage of Messenger among youth is verified. It remains to be seen if it is possible. At least with RiveScript the framework is easily copied and modified.

4.5 Alternative Technical Solutions

There are two Finnish products, which are good alternatives for RiveScript. GetJenny is a chatbot platform, which also has an open source -version. Giosg is an online SaaS-service, which offers their clients a platform where to chat or use chatbots. More about these in the chapters below.

4.5.1 GetJenny

GetJenny is a “multilingual chatbot platform”, which is said to be able to automate 40-80 % of the customer support chats after just a few months of usage [71]. GetJenny provides also wider-scale AI (NLP, Machine Learning). The service is divided in three layers: the actual chatbot taking care of the repetitive questions, Smartlayer, which can be trained with the client’s own data, and Web Application, which is the user interface for launching, editing and teaching chatbots. GetJenny is a paid service: its pricing begins from 990 euros per month, but with 2000 euros per month “Metropolia could go far” [72].

GetJenny offers also an open source repository of their service [73]. This open source chatbot, StarChat, is targeted for business-to-business (B2B) applications, it is a scalable conversational engine, which’s code is in GetJenny’s github repository. StarChat uses Elasticsearch and Java, setup is recommended to be done with Docker. A good example of a GetJenny chatbot is Emma, which insurance company IF launched in March 2017. At the beginning, Emma was taught to answer 50 questions of frequently asked topics. After 6 months, it was handling 250 topics and 60 % of the customer service enquiries. [74]

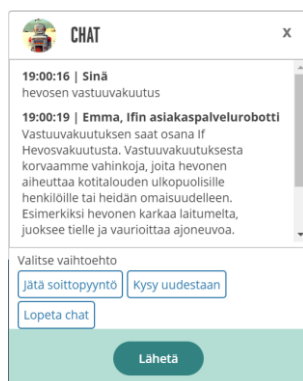


Figure 28. Emma, example of a chatbot built on GetJenny.

4.5.2 Giosg

It is said, that the idea of Giosg came from kiosk: one can get the services one needs from one kiosk. Giosg is an analytics platform where a chat is one of the services. [75] Taking Giosg in use to Metropolia has been under examination and discussion for a long time, it would serve users throughout Metropolia. Giosg doesn't offer a chatbot straight but can offer it via subcontractors: Giosg have used at least GetJenny and Ultimate.ai.

The costs for Giosg's services are for example with one username (multiple users can use, though not simultaneously), 10 rules and limited use of callback form 150 euros per month. An extra user can be bought with 100 euros per month. [75] The chatbot costs quite a lot more per month, as mentioned in the chapter above. Still, this option is worth considering, since nothing is free and there is always the need for human resources in any solution.

5 Discussions and Conclusions

Sounds like a cliché, but the world is changing. People are facing a change that could be compared to industrial revolution in the turn of 1800s. Many forms and structures are experiencing changes which Machine Learning, AI and chatbots are bringing. The ways we process things, issues and life is going to change.

The chatbots have been appearing to business in the form of online assistants in corporate web sites. In education sector this has been noticed too, and those brave-ones who have dared to set up a chatbot enjoy the benefits. The youth is mobile-oriented and appreciate IM in everything - assuming, that they get the response right away. So, how self-evident the resolution actually is? Chatbots are always ready to respond immediately, to dispense relevant, current information. They can also boost for example enrolments to courses or play the role of an e-campus-guide. In addition, chatbots gather chat history, which could give valuable information of what issues are asked and when. [76]

The main trigger for doing this thesis was to create a proof of concept of a chatbot, which would serve the students of Metropolia UAS. The client ordering this service was Metropolia UAS's Student Affairs Office. By this chatbot the client hoped, that it would help reducing the customer servants' workload in answering to students' routine questions and at the same time serve the students better. The goal was to include a demo to the proof of concept by using such a solution, with which it would be easy to continue to the actual implementation.

The requirements for the technical solution were Finnish as operating language, free of charge, possibility that the chatbot could direct questions to customer servants (at least by providing an email link) and that the chatbot's answer could include a clickable link. One requirement was also that the chatbot could be planted in Metropolia's servers. This requirement came up when it was decided that the chatbot would be planted to Metropolia's intra, to Student's Desktop.

"Finnish as operating language" and especially "free of charge" were the requirements which narrowed the real options down to a few. JavaScript library and framework RiveScript was the only totally free one and it also met all the other requirements too. Rivescript was also possible to be used by a not-so-experienced-coder, so it was chosen. Free of charge wasn't the most important requirement but Finnish as operating language was, so GetJenny and a company offering it and Ultimate.ai, Giosg, felt good ones to be

brought up as alternative options. GetJenny and Ultimate.ai include also wider-scale AI (NLP, Machine Learning), but perhaps even Watson or Luis.ai would have been chosen if there would have been a big budget.

This proof of concept includes designing a chatbot as a whole: interaction and appearance design, recommendations about the technical implementation and also a small demo, as mentioned. Designing a chatbot match to web design in some extent, but still it took a while to get into the heart of chatbots' world. The main finding was that adapting and setting up a RiveScript framework was kind of a head ache when doing such things for the first time. Finding out what ever code snippet is going to what file and which command is run in the Command Prompt and which somewhere else was sometimes overwhelming. But it was worth it! Luckily the actual syntax was easy and simple to modify to this precise use.

So, RiveScript worked well for the demo, the case of this thesis. Machine Learning and NLP had to be excluded from this demo, since they didn't fit to the scope of a thesis. The demo worked also without them, but in larger use it isn't efficient. With NLP the difficulty with Finnish language (word forms) is possible to overcome more efficiently and also the interaction is possible to get more conversational. Maybe another student or a funded project could enhance the chatbot and add NLP and/or Machine Learning to it.

Year 2018 was said to be the year of the chatbots and in order to make a right-on-time appearance in this field, now is the time to act. DNA has forecasted, that in year 2018 chatbots start to speak Finnish and the most popular platform is going to be Facebook Messenger [77]. The chatbot of this thesis wasn't created on Messenger because the chatbot was decided to put in Metropolia's intra, OMA, and it can also be questioned whether the youth uses Facebook and Messenger so much. In addition, also Gartner forecasts good to chatbots, it says even, that bots will take over. Gartner says that by 2021, more than half of the enterprises will spend more on chatbot creation than mobile app development. Chatbots will become the face of AI. [78] To implement a chatbot to higher education is very modern and acting as a forerunner of development. Now is the time to act, if one wishes to be remembered on this topic!

6 Summary

This thesis was about creating a proof of concept of a chatbot to Metropolia UAS Student Affairs Office by using a ready-made technical solution.

The research questions were:

- Which of the existing technical solutions could be used in order to get a chatbot serve the students of a UAS?
- How to design a chatbot to UAS? What kind of modifying and raw work is needed to get a well-implemented chatbot to UAS?

The main finding about technical solutions was that when aiming for Finnish as operating language, free of charge, possibility to direct questions to customer servants and handling links as answers to user's questions, there aren't that many options. An open source JavaScript library and framework called RiveScript was the best-suited solution for this case's requirements. It was the best one also from the builder point of view: it has easy-to-learn syntax, it is simple to use, and it can be modified to match one's needs. Even though for example Facebook Messenger is very popular, it wasn't suitable in this case, since the chatbot was wanted to put in Metropolia's intra, OMA.

This thesis presents the history and theory of chatbots, shows the process of designing a chatbot and displays how to use a ready framework to modify an own chatbot demo. Since the chatbot was actually planned to be put into service, this thesis contains recommendations of how to design and create it and also what needs to be considered in order to make the chatbot work better. In addition, this thesis includes plans and ideas how to test, market and release the chatbot. Since this chatbot was asked and planned to work more like a FAQ, all the technical modifying of the demo focused on fetching and presenting the answer and only just a little to fluent conversation. That is something to develop further.

Machine Learning and NLP didn't fit to the scope of a thesis but the demo worked also without them. However, it was obvious, that at least NLP is needed to the actual chatbot. When dealing with Finnish language and its word forms, NLP is quite crucial from the efficiency point of view. The demo gave high hopes that also the actual chatbot could be made by using these techniques.

This experiment was small, but it sheds light on how a chatbot could be made, tested and released so that it could work in UAS-sector. This kind of simple, routine answering to basic questions which this demo presented, can be outsourced to chatbots in multiple fields and areas, so why not also in higher education. A chatbot is nice and easy and serves both sides, the students and the UAS, namely its customer servants.

Perhaps the limit of chatbots goes in social skills and mental health: chatbots can handle a lot of basic issues but at least for now emotions are out of their reach. It has been predicted, that the chatbots will come more and more in use and they develop further continuously. Five years ago, chatbots were said to be the future of higher education, now they are here to stay [79]. For example, a company called VirtualSpirits titled itself to be the best chatbot platform provider for universities and higher education, and it's powering over 12 000 chatbots in all sectors [80]. Chatbots will change the world by changing the everyday life of people. Maybe someday all universities have their own chatbot guiding their students and giving advices, motherly herding them to the sources of information and help - without patronizing. Sounds feasible.

References

- 1 Usability.gov. User-Centered Design Basics. Web article. <<https://www.usability.gov/what-and-why/user-centered-design.html>>. Accessed 27 April 2018.
- 2 Wikipedia. Online chat. Web article. <https://en.wikipedia.org/wiki/Online_chat>. Accessed 27 Dec 2018.
- 3 Mauldin M., ChatterBots, TinyMuds, and the Turing Test: Entering the Loebner Prize Competition. AAAI Press. 1994.
- 4 Chatbots Magazine. Chatbot Design Trends 2018. Web article. <<https://chatbotsmagazine.com/chatbot-design-trends-2018-253fb356d3a3>>. Accessed 28 Dec 2017.
- 5 Turing A., Computing Machinery and Intelligence. Oxford University Press, Mind, Volume LIX, Issue 236, pages 433–460. 1950.
- 6 Harnad S., The Annotation Game: On Turing (1950) on Computing, Machinery, and Intelligence, in Epstein, Robert; Peters, Grace, The Turing Test Sourcebook: Philosophical and Methodological Issues in the Quest for the Thinking Computer. Kluwer. 2008.
- 7 Wikipedia. Turing Test. Web article. <https://en.wikipedia.org/wiki/Turing_test>. Accessed 3 Jan 2018.
- 8 The Harvard Gazette. Alan Turing at 100. Web article. <<https://news.harvard.edu/gazette/story/2012/09/alan-turing-at-100/>>. Accessed 3 Jan 2018.
- 9 Wikipedia. ELIZA. Web article. <<https://en.wikipedia.org/wiki/ELIZA>>. Accessed 4 Jan 2018.
- 10 Weizenbaum J., ELIZA - A Computer Program for the Study of Natural Language Communication between Man and Machine. Communications of the ACM. 1966.
- 11 Wikipedia. Chatbot. Web article. <<https://en.wikipedia.org/wiki/Chatbot>>. Accessed 5 Jan 2018.
- 12 Wikipedia. Kalle Kotipsykiatri. Web article. <https://fi.wikipedia.org/wiki/Kalle_Kotipsykiatri>. Accessed 5 Jan 2018.
- 13 Wikipedia. Weak AI. Web article. <https://en.wikipedia.org/wiki/Weak_AI>. Accessed 7 Jan 2018.
- 14 Wikipedia. Eugene Goostman. Web article. <https://en.wikipedia.org/wiki/Eugene_Goostman>. Accessed 5 Jan 2018.
- 15 Microsoft. Add bots for private chats and channels in Microsoft Teams. Web article. <<https://docs.microsoft.com/en-us/microsoftteams/add-bots>>. Accessed 6 Jan 2018.

- 16 Techtarget. IM-bot. Web article. <<http://searchdomino.techtarget.com/definition/IM-bot>>. Accessed 6 Jan 2018.
- 17 Luger G.F, Artificial Intelligence: Structures and Strategies for Complex Problem Solving. Fourth edition. Addison-Wesley. 2002.
- 18 Techopedia. Weak Artificial Intelligence. Web article. <<https://www.techopedia.com/definition/31621/weak-artificial-intelligence-weak-ai>>. Accessed 7 Jan 2018.
- 19 Negnevitsky M., Artificial Intelligence: A Guide to Intelligent Systems. Addison-Wesley. 2002.
- 20 Business Insider Inc: Microsoft deletes racist genocidal tweets from AI chatbot Tay. Web article. <<http://www.businessinsider.com/microsoft-deletes-racist-genocidal-tweets-from-ai-chatbot-tay-2016-3?r=US&IR=T&IR=T>>. Accessed 9 April 2018.
- 21 Deutsche Welle Business: How Microsoft's Chatbot Learned to be a Jerk. Web article. <<https://www.dw.com/en/ho-microsofts-chatbot-learned-to-be-a-jerk/a-19142538>>. Accessed 9 April 2018.
- 22 Bostrom N., Superintelligence: Paths, Dangers, Strategies. Oxford University Press. 2016.
- 23 Maxwell, J.C.: "Does the Progress of Physical Science tend to give any Advantage to the Opinion of Necessity (or Determinism) over that of the Contingency of Events and the Freedom of the Will?" in: L. Champbell/ W. Garnett: "The Life of James Clerk Maxwell". London 1882, S. 440ff.
- 24 Wikipedia. Technological Singularity. Web article. <https://en.wikipedia.org/wiki/Technological_singularity>. Accessed 11 March 2018.
- 25 Suomi 2050, theme appendix of Ilta-Sanomat in December 2017.
- 26 Nordea News. She is the Company's Most Effective Employee. Web article. <<https://nordeanews.no/2017/09/hun-er-bankens-mest-effektive-medarbeider/>>. Accessed 30 Jan 2018.
- 27 VentureBeat. Better Believe the Bot Boom is Blowing up Big for B2B, B2C Businesses. Web article. <<https://venturebeat.com/2016/07/24/better-believe-the-bot-boom-is-blowing-up-big-for-b2b-b2c-businesses-vb-live/>>. Accessed 12 Jan 2018.
- 28 Chatbot News Daily. Chatbots Take Education to the Next Level. Web article. <<https://chatbotnewsdaily.com/chatbots-take-education-to-the-next-level-23bc02cdbccf>>. Accessed 14 Feb 2018.
- 29 VentureBeat. Facebook Opens Its Messenger Platform to Chatbots. Web article. <<https://venturebeat.com/2016/04/12/facebook-opens-its-messenger-platform-to-chatbots/>>. Accessed 17 Jan 2018.
- 30 Rocketbots. Does Your Business Need a Chatbot? Web article. <<https://rocketbots.io/blog/does-your-business-need-a-chatbot/>>. Accessed 19 Jan 2018.

- 31 Virtual Agent Chat. Conversational Toys - The Latest Trend in Speech Technology. Web article. <<https://virtualagentchat.com/2015/02/23/conversational-toys-the-latest-trend-in-speech-technology/>>. Accessed 20 Jan 2018.
- 32 Fast Company. NAGY, EVIE. Using Toytalk Technology, New Hello Barbie Will Have Real Conversations With Kids. Web article. <<http://www.fastcompany.com/3042430/most-creative-people/using-toytalk-technology-new-hello-barbie-will-have-real-conversations>>. Accessed 28 Jan 2018.
- 33 Twit. Oren Jacob, the co-founder and CEO of ToyTalk Interviewed on the TV-show Triangulation on the TWiT.tv Network. Web interview. <<https://twit.tv/shows/triangulation/episodes/179>>. Accessed 28 Jan 2018.
- 34 Artificial Intelligence Script Tool. Web page. <<https://patents.google.com/patent/US20140032471> >. Accessed 1 Feb 2018.
- 35 Chatbots Magazine. Be Prepared for 2018 the Year of the Bot. Web article. <<https://chatbotsmagazine.com/be-prepared-for-2018-the-year-of-the-bot-c9f80516b844>>. Accessed 1 Feb 2018.
- 36 Wikipedia. Natural-language Processing. Web article. <https://en.wikipedia.org/wiki/Natural-language_processing>. Accessed 3 Feb 2018.
- 37 Chatbots Magazine. The Ultimate Guide to Designing a Chatbot Tech Stack. Web article. <<https://chatbotsmagazine.com/the-ultimate-guide-to-designing-a-chatbot-tech-stack-333eceb431da>>. Accessed 3 April 2018.
- 38 Natural Language Toolkit. NLTK 3.3 Documentation. Web page. <http://www.nltk.org/_images/tree.gif>. Accessed 3 Feb 2018.
- 39 Wikipedia. Morphology (linguistics). Web article. <[https://en.wikipedia.org/wiki/Morphology_\(linguistics\)](https://en.wikipedia.org/wiki/Morphology_(linguistics))>. Accessed Sep 11 2018.
- 40 Lingsoft. FINTWOL. Web service. <<http://www2.lingsoft.fi/cgi-bin/fintwol>>. Accessed Sep 11 2018.
- 41 Wikipedia. Machine Learning. Web article. <https://en.wikipedia.org/wiki/Machine_learning>. Accessed 3 Feb 2018.
- 42 Wikipedia. Sentiment Analysis. Web article. <https://en.wikipedia.org/wiki/Sentiment_analysis>. Accessed 6 Feb 2018.
- 43 Chatbots Magazine. Chatbot Report 2018: Global Trends and Analysis. Web article. <<https://chatbotsmagazine.com/chatbot-report-2018-global-trends-and-analysis-4d8bbe4d924b>>. Accessed Sep 11 2018.
- 44 Chatbots Magazine. The Complete Beginner's Guide to Chatbots. Web article. <<https://chatbotsmagazine.com/the-complete-beginner-s-guide-to-chatbots-8280b7b906ca>>. Accessed 9 Feb 2018.
- 45 Giosg. How Does a Chatbot Work. Web article. <<https://www.giosg.com/find-answers/how-does-a-chatbot-work>>. Accessed 11 Feb 2018.

- 46 YouTube. Make a Facebook Chatbot from Scratch in 20 Minutes | BOTS | Quick Code. Web video. <<https://www.youtube.com/watch?v=bUwiKFTvmDQ>>. Accessed 11 Feb 2018.
- 47 UX Collective. Designing Chatbots: A Step-by-Step Guide with Example. Web article. <<https://uxdesign.cc/how-to-design-a-robust-chatbot-interaction-8bb6dfae34fb>>. Accessed 12 Feb 2018.
- 48 Chatbots Magazine. The Ultimate Cheat Sheet for Building the Best Chatbot. Web article. <<https://chatbotsmagazine.com/the-ultimate-cheat-sheet-for-building-the-best-chatbot-2e7ab821d3a3>>. Accessed 12 Feb 2018.
- 49 Intercom. Principles of Bot Design. Web article. <<https://blog.intercom.com/principles-bot-design/>>. Accessed 15 Feb 2018.
- 50 Medium. Principles of Conversational Design. Web article. <<https://medium.com/@LinusEkenstam/principles-of-conversational-design-c4778e620201>>. Accessed 16 Feb 2018.
- 51 Usability Body of Knowledge. Web page. <<http://www.usabilitybok.org/glossary>> Accessed 16 April 2018.
- 52 Giosg: Chatbots in customer service - 6 tips - From IF Insurance to Successful Implementation. Web document. <<https://www.giosg.com/6-tips-from-if-insurance-to-successful-implementation-of-chatbots>>. Accessed 19 March 2018.
- 53 Giosg. How to Ensure Excellent Customer Experience. Web article. <<https://www.giosg.com/find-answers/how-to-ensure-excellent-customer-experience-with-chatbots>>. Accessed 23 March 2018.
- 54 Giosg. How Does a Chatbot Learn. Web article. <<https://www.giosg.com/find-answers/how-does-a-chatbot-learn>>. Accessed 3 March 2018.
- 55 Giosg. How to be Sure the Chatbot Replies Correctly in Conversations. Web article. <<https://www.giosg.com/find-answers/how-to-be-sure-the-chatbot-replies-correctly-in-conversations>>. Accessed 7 March 2018.
- 56 Yuan M., Chatbots: Building Intelligent, Cross-Platform, Messaging Bots. Addison-Wesley. 2016.
- 57 VentureBeat. Oracle Launches a Chatbot Development Platform. Web article. <<https://venturebeat.com/2016/09/18/oracle-launches-a-chatbot-development-platform/>>. Accessed 7 March 2018.
- 58 Oracle. Oracle Introduces AI-Powered Intelligent Bots to Help Enterprises Engage Customers and Employees. Web article. <<https://www.oracle.com/corporate/pressrelease/ow17-ai-powered-intelligent-bots-100217.html>>. Accessed 9 March 2018.
- 59 Oracle Cloud. Mobile Service. Web article. <<https://cloud.oracle.com/mobile/features>>. Accessed 10 March 2018.
- 60 Telerik Developer Network. What is a Webview. Web article. <<https://developer.telerik.com/featured/what-is-a-webview/>>. Accessed 12 April 2018.

- 61 Rivescript. Web page. <<https://www.rivescript.com/>>. Accessed 2 June 2018.
- 62 Usability Body of Knowledge. Web article. <<http://www.usabilitybok.org/what-is-usability>>. Accessed 15 April 2018.
- 63 Nielsen, J., Usability 101: Introduction to Usability. Web article. <<https://www.nngroup.com/articles/usability-101-introduction-to-usability/>>. Accessed 15 April 2018.
- 64 Interaction Design Foundation. 7 Great, Tried and Tested UX Research Techniques. Web page. <<https://www.interaction-design.org/literature/article/7-great-tried-and-tested-ux-research-techniques>>. Accessed 15 June 2018.
- 65 Botsociety. How to Perform User Testing on Your Chatbot. Web article. <<https://botsociety.io/blog/2018/02/user-testing-chatbot/>>. Accessed 5 July 2018.
- 66 Chatbots Magazine. How to Test a Chatbot Part 1: Why is it So Hard. Web article. <<https://chatbotsmagazine.com/how-to-test-a-chatbot-part-1-why-is-it-so-hard-10f1ee8ca37d>>. Accessed 5 July 2018.
- 67 Varma. Eipäs kiroilla tuhmeliini. Web article. <<https://www.varma.fi/muut/uutishuone/uutiset/2018-q1/eipas-kiroilla-tuhmeliini-varman-chattibotti-helmi-auttaa-asiakkaita-247/>>. Accessed 8 July 2018.
- 68 RiveScript. Tutorial. Web page. <<https://www.rivescript.com/docs/tutorial>>. Accessed 2 March 2018.
- 69 RiveScript. RiveScript Interpreters. Web page. <<https://www.rivescript.com/interpreters>>. Accessed 2 March 2018.
- 70 GitHub. Aichaos/rivescript-js. Repository. <<https://github.com/aichaos/rivescript-js>>. Accessed 8 March 2018.
- 71 GetJenny. Frequently asked questions. Web page. <<https://www.getjenny.com/faq>>. Accessed 28 August 2018.
- 72 Email interview with Ilkka Vertanen, Sales Director at GetJenny. 28 August 2018.
- 73 StarChat. StarChat Documentation. Web page. <<https://getjenny.github.io/starchat-doc/>>. Accessed 28 August 2018.
- 74 If and GetJenny case study. Web document. Available for downloading when ordered. <<https://www.getjenny.com/case-study-if?hsCtaTracking=3d6fad8e-e881-4e85-a547-84858412c5db%7Cfe3247d0-171b-48e4-acaf-4069a3e281d8>>. Accessed 12 Sep 2018.
- 75 Meeting with Kalle Mäkelä, Nordics Sales Director at Giosg. 24 April 2018.
- 76 Chatbotslife. Chatbot for University – 4 Challenges Facing Higher Education and How Chatbots Can Solve Them. Web article. <<https://chatbotslife.com/chatbot-for-university-4-challenges-facing-higher-education-and-how-chatbots-can-solve-them-90f9dcb34822>>. Accessed 13 Sep 2018.

- 77 DNA. Miltä Näyttää Teknologiavuosi 2018. Web article. <https://www.dna.fi/yrityksille/blogi/-/blogs/milta-nayttaa-teknologiavuosi-2018-?utm_source=dnabusiness&utm_medium=email&utm_campaign=sa_kaikki&utm_content=mitla-nayttaa-teknologiavuosi-2018-&ircp=116289074>. Accessed 13 Feb 2018.
- 78 Gartner. Gartner Top Strategic Predictions for 2018. Web article. <<https://www.gartner.com/smarterwithgartner/gartner-top-strategic-predictions-for-2018-and-beyond/>>. Accessed 10 Feb 2018.
- 79 Chatbotslife. Higher Education Chatbot: Chatbots Are the Future of Higher Education. Web article. <<https://chatbotslife.com/higher-education-chatbot-chatbots-are-the-future-of-higher-education-51f151e93b02>>. Accessed 13 Sep 2018.
- 80 VirtualSpirits. Chatbot for Universities. Web site. <<https://www.virtualspirits.com/chatbot-for-university.aspx>>. Accessed 13 Sep 2018.