

UNIVERSITY OF TARTU  
Institute of Computer Science  
Computer Science Curriculum

Jan Moppel

# Socratic chatbot

Bachelor's Thesis (9 ECTS)

Supervisor: Taivo Pungas, MSc  
Co-supervisor: Sven Aller, MSc

Tartu 2018

## **Socratic chatbot**

**Abstract:** Conversational systems become more popular with each day. They are still cannot communicate with humans like other humans, but with the development of technologies chatbots become an inherent part of our life. Applying new methods of communication is very important for their progress and this work is reviewing the possibility of using the Socratic questioning method on the conversational systems.

**Keywords:**

conversational system, chatbot, socratic method, natural language processing

**CERCS:** P170 - Computer science, numerical analysis, systems, control

## **Sokraatiline juturobot**

**Lühikokkuvõte:** Vestlussüsteemid muutuvad iga päevaga üha populaarsemaks. Nad ei suuda veel suhelda inimestega sarnaselt teiste inimestega, kuid tehnoloogia arenguga saavad juturobotid tulevikus meie elu lahutamatuks osaks. Uute kommunikatsioonimeetodite rakendamine on nende arengu jaoks väga tähtis. Selles töös vaadeldakse sokraatilise meetodi kasutusvõimalusi vestlussüsteemides.

**Võtmesõnad:**

vestlemissüsteem, juturobot, sokraatiline meetod, keeletehnoloogia

**CERCS:** P170 - Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
<b>2</b>	<b>Overview</b>	<b>6</b>
2.1	Socratic Method . . . . .	6
2.2	Chatbot . . . . .	6
2.2.1	Conversational systems today . . . . .	6
2.2.2	Chatbots using the Socratic Method . . . . .	7
2.3	Elizabeth . . . . .	8
<b>3</b>	<b>Design</b>	<b>10</b>
3.1	Conversation rules . . . . .	10
3.2	Applying the Socratic Method . . . . .	10
3.3	Conversations. . . . .	11
3.3.1	Dialogues setup. . . . .	11
3.3.2	Conversation example. . . . .	12
3.3.3	Conversations output. . . . .	15
<b>4</b>	<b>Implementation</b>	<b>16</b>
4.1	Modules and libraries . . . . .	16
4.2	User input analyzing and Response generating . . . . .	17
<b>5</b>	<b>Validation</b>	<b>20</b>
5.1	First iteration . . . . .	20
5.1.1	Experiment setup . . . . .	20
5.1.2	Feedback . . . . .	20
5.1.3	Changes . . . . .	21
5.2	Second iteration . . . . .	22
5.2.1	Experiment setup . . . . .	22
5.2.2	Feedback . . . . .	22
5.2.3	Results . . . . .	26
5.3	Future development . . . . .	27
<b>6</b>	<b>Conclusion</b>	<b>28</b>
	<b>References</b>	<b>31</b>
	I. Licence . . . . .	32

# 1 Introduction

Sometimes people face a problem, which they don't know how to solve or even not sure that it is a problem. In both cases, they try to describe it to others to get advice. Moreover, very often, it happens that people find a solution without even receiving a suggestion. They just analyze their issue and find an answer that they needed. The analytical procedure in this case is stimulated by clarification questions. The interlocutor or the person himself can ask them. Frequently these questions do not require any kind of additional knowledge except for information that is used in the current dialogue. This critical-thinking stimulation method is called the Socratic Method.

It is used in different areas, such as teaching [1], psychotherapy [2] etc. In theory, the Socratic Method makes possible to use a conversational system or a chatbot to better understand the issue and solve it, without other person's help.

There were some attempts to bring this idea to life. For example, one of the first conversational systems was ELIZA [3]. It simulated a Rogerian psychotherapist and responded with non-directional questions to user inputs. Whereas ELIZA could keep up a conversation, its technical solution (pattern matching without input's saving) makes it impossible to sustain a long-conversation with specific clarification questions. Another good example is a Woebot [4]. It makes user to analyze himself by asking general and clarification questions about the mood, activities, etc. Although, Woebot uses more advanced Natural Language Processing techniques, it cannot sustain a long conversation and ask specific questions based on user input.

Right now Artificial Intelligence and Conversational Systems are very widespread and continue to grow in popularity. Siri [5], Amazon Alexa [6], Google Assistant [7] can do different tasks for human. Besides that, with the development of messaging platforms and their popularity growth (in 2015 their usage has surpassed the social media usage) [8], chatbot integration became very easy. Most popular platform for bots implementation is Facebook Messenger: 100 000 bots (April 2017) [9]. The biggest popularity chatbots got in the business field: by 2020, over 80% of businesses are expected to have some sort of chatbot automation implemented [10]. They help to personalize the customer experience and minimize cost on customer support [11].

In such a way, conversational systems will continue to become an inherent part of everyday life and a bot that would make a problem-solving process faster and easier would be a very useful assistant. The goal of this thesis is to explore chatbots, Socratic Method and try to create a conversational system that will use this method. Consequently, this conversational system's title is the Socratic chatbot [12].

If this system is able to memorize, extract useful information from the user input, ask adequate questions based on this information, carry dialogue and make user rethink his problem at least in some cases, then the work may be considered as successful.

To achieve these results, some restrictions should be defined. The main one is based on the Socratic Method itself: user can only write statements, bot can only ask questions.

The user also has to answer to them accurately and not too broad. It would make input understanding much easier and achievable.

This thesis consists of 4 parts. The first part - **Design** - gives an overview about the vision of this project: examples of dialogues, which are considered as a desired result; representation of bot's behavior strategy. The second part - **Implementation** - is a detailed building process description: what modules and libraries are used; how user input analyzing goes and response generating processes. The third part - **Validation** - gives an overview about the validation process: who and how tested the solution; what was the feedback; how the system was changed during this process and what are the results. The last part - **Conclusion** - summarizes the thesis.

## 2 Overview

### 2.1 Socratic Method

As written below, the Socratic Method is an irreplaceable tool for developing critical thinking. The goal of this method may be finding the truth, teaching new concepts, helping in deeper understanding of things, etc. The structure [13] of it may be formulated like this:

1. Define common sense statement;
2. Find an exception to that statement;
3. If the exception is found - reformulate the statement according to it;
4. Repeat the cycle until the exception cannot be found.

This algorithm can be reproduced both in dialogue with somebody and without anyone.

Sometimes people specify two main types of this tool [14]: classic and modern. The classic Socratic Method is used to find an answer to questions, which don't have a clear answer. For instance, questions like "What is love?", "What is justice?" etc. Finding a clear answer (statement) to these everlasting inquiries, which won't have any exceptions, is impossible. A good example of using the classic Socratic Method is the dialog between preacher and "Socrates" about the essence of the moral person [15]. This approach forms the basis of cognitive therapy [16]. By using it, patient and doctor deepen into the problem and start to understand it, which helps to define the treatment.

The modern Socratic Method is more applicable in everyday life and conversations. Its goal is not to discover the truth, but to lead the interlocutor to its understanding. In other words, it is used, when one of the dialog partners already knows the answer, but the other one doesn't. This approach is very widespread in teaching. For example, geometry experiment in Plato's "Meno" [17], Garlikov's experiment [1], etc.

### 2.2 Chatbot

Chatbot or conversational system is "a computer program designed to simulate conversation with human users, especially over the Internet." [18] Conversations are carried on via textual or auditory methods.

#### 2.2.1 Conversational systems today

Different scales can classify chatbots: implementation design, domain usage, tasks and goals and so on [19]. For example, by implementation design the Google Assistant [7] is an intelligent agent, because it doesn't require human intervention to correctly perform

its routines [20]. By domain usage, it is a generalist bot, because depending on the user desire, it can do very different tasks from playing music to telling the weather forecast. In addition, by tasks and goals, Google Assistant can be classified as a chatbot, because it can sustain conversations. At the same time, it can provide some requested information and play different games with user, which makes it also an informational and entertainment bot. So consequently, bots can be classified in various manners depending on selected scale.

As mentioned in the introduction, the popularity of chatbots is growing. Primarily this is due to simplicity of building them for concrete business. Now, not only big companies such as Uber [21] or CNN [22] use chatbots, but also even for small businesses it became accessible because of platforms for building conversational systems. Dialogflow [23], Wit.ai [24] and other platforms make the implementation process fast and easy.

To build a simple chatbot using these platforms, user need to add some text there and manually choose specific parts of this input, that bot should be looking for. These parts are called *entities*. Then user defines what these entities mean. In other words, he names them. It can be a very new name or already existing one from the platform. Finally, user defines action needs to be performed, when bot recognizes these named entities during the conversation.

In general, by implementation design these bots are classified as scripted, because their interactions are based on scripts (pre-determined models), and as specialists by domain usage classification, which means that they have narrow-minded functionality for specific tasks. For example, there is a pizza delivery bot and user inserts: "I want to order big Margarita pizza". This chatbot understands, what does user wants, by recognizing three predefined entities: intent ("order"), size ("big") and title ("Margarita"). Then system can act according to it.

The biggest companies in IT field are also working on their virtual assistants, but they are focused on creating an intelligent agent and generalist bot. For example, already mentioned Alexa by Amazon [6] and Google Assistant [7], Cortana by Microsoft [25], etc. These systems can perform many different tasks from ordering items from the internet to finding the closest restaurant, converse with the user about different subjects and so on.

### **2.2.2 Chatbots using the Socratic Method**

Talking about the Socratic Method, there are self-help chatbots that use it in some ways. For example, Woebot [4] and X2 AI (Tess) [26]. They both are therapist-bots and use the cognitive-behavioral therapy (CBT). As already mentioned above, CBT is based on the Socratic Method [16].

Woebot and Tess aim to help users suffering from depression and anxiety through talking about interlocutor's mood and feelings. Tess uses only text-based communication. Woebot, on the other hand, very often uses pictures and buttons with predefined answers

in the dialogues [27].

However, the Socratic chatbot is more like a companion, not a therapist-bot. Therefore, the closest to it and the most well-known chatbot that can be used to help users better understand their problems, remains Elizabeth.

## 2.3 Elizabeth

Elizabeth [28] is an interpretation of Joseph Weizenbaum's original conversational system ELIZA [3]. It has the same personality (Rogerian psychotherapist, who is always asking questions) and uses the same technique (scripts and pattern matching) for input understanding and output generating, but Elizabeth's approach is more advanced, which leads to increased performance. For example, in distinction to ELIZA, Elizabeth is able to change its scripts, if the interlocutor continuously uses language patterns. It is an example of scripted, but generalist chatbot. The bot's response generating process consists of 4 steps (see Figure 1):

1. Text preprocessing;
2. Pattern matching;
3. Dynamic processing;
4. Grammatical rules implementation.

On the first step, Elizabeth transforms text into required form. User input is converted to lower case, tokenized and cleaned from unnecessary characters, except for: ! " ' ( ) , - . 0...9 : ; ? A...Z a...z.

On the second step, initial response is generated. Firstly, the system generalizes user input by applying input transformation rules. E.g. converting words "mum" and "dad" into "mother" and "father" respectively. Secondly, Elizabeth tries to find keywords in the input. If the text contains a keyword, then initial response will be selected according to it. Otherwise, initial response will be randomly selected from the list of broad predefined sentences made for this type of case. If the response is not static, output-transforming rules are applied to it. Finally, the response is uppercased.

On the third step, the system updates the script file with all the rules according to the previous step. It adds, removes and memorizes script commands (rules).

On the last step, Elizabeth corrects some grammatical mistakes that may appear during response generating. Grammatical rules are also predefined. E.g., word "readed" converts into "red".

Elizabeth (ELIZA) and Socratic chatbot have a lot in common. E.g., both ask clarification questions that are based on the user input, both can memorize user answer to use it later and can help people rethink their problems. Nevertheless, Elizabeth doesn't



The following trace using the previous script to illustrate different inputs with its responds:

User Input	I nput Transforming	Keyword Patterns	Output Transforming	Respond	Actions
Dad loves Mum	Match: '001' mum => mother '002' dad => father	Match: '001' K MOTHER '001' R TELL ME MORE ABOUT YOUR FAMILY.	-	TELL ME MORE ABOUT YOUR FAMILY.	-
My sister is a teacher.	-	Match: '006' K my [phrase] [phrase] => sister is a teacher '001' R YOUR is a teacher?	-	YOUR SISTER IS A TEACHER?	'fam' M sister is a teacher
My brother is younger than me.	-	Match: '004' K [phrs1] IS YOUNGER THAN [phrs2] [phrs1] => my brother [phrs2] => me '001' R SO [phrs2] IS OLDER THAN [phrs1].	Match: [phrs1] with '003' my => YOUR [phrs2] with '005' me => YOU	SO YOU IS OLDER THAN YOUR BROTHER.	-
I like reading	-	Match '005' K I LIKE [string]ING [string] => read '001' R HAVE YOU [string]ED AT ALL RECENTLY?	-	HAVE YOU READED AT ALL RECENTLY?	-

Figure 1. Elizabeth working process according to the script. [28]

have any clear end goal in the communication. It just reacts to the user input, and no reactions are connected with the flow of conversation. Elizabeth also uses only pattern matching as a main technique, which makes its potential for the future development very low.

However, the specific underlying method with well-defined algorithm in it (classic Socratic Method), modern technologies (dependency tree parsing, sentiment analysis, named entities recognition, automatic grammar correction) and clear focus on helping people better understand their problems through activating critical-thinking, is what drastically distinguish the Socratic chatbot from any other conversational system.

## 3 Design

The design of any chatbot is very important, because it defines the implementation and expectations from the final result. How this conversational system will be used, what personality it has and how it will communicate with users? All these questions are addressed during the design stage.

### 3.1 Conversation rules

As already mentioned above, the main difference between the Socratic chatbot and other conversational systems is the direct use of the Socratic Method. In order to apply its classic approach effectively, some general rules for the bot needed to be specified:

1. Bot starts the conversation;
2. Bot can only ask questions;
3. User can only answer them;
4. Questions are based on the user input only;
5. User answers should not be too long or broad;
6. The conversation subject doesn't have any limits;
7. Conversation continues until user stops responding.

Applying these principles empowers to build human-bot communication based on the Socratic Method, which is understandable for user and obtainable for conversational system.

### 3.2 Applying the Socratic Method

The conversation itself is built according to the method's structure, but with a more practical approach. The dialogue for the chatbot consists of 6 steps or stages:

1. **Get user statement.** User defines the problem to discuss with the bot.
2. **Get Reasons.** Bot asks about reasons of the statement. Mostly used in the beginning.
3. **Get Past Experience.** Bot asks about past experience, using every received reason individually.

4. **Get Exception.** Considering available data, bot asks about possible exception in the statement.
5. **Control.** Check whether all available data should be stored (relevant).
6. **Finish.** User decides to finish the dialogue.

Out of every user input, depending on the stage, bot extracts valuable information and uses it in its questions. Each step has its own set of predefined questions or patterns, but the filling of the question depends on the stage itself and available data. The Socratic chatbot continuously repeats these stages, according to available data, until user won't finish the conversation.

### 3.3 Conversations.

To better understand, how bot should use this algorithm to communicate, several dialogues were created.

#### 3.3.1 Dialogues setup.

These dialogues had to represent the ideal cases of using the Socratic chatbot. In order to do this, every communication was made according to these rules, which are similar to the *Wizard of Oz* simulating method [29]:

1. 2 people participated in each dialogue, where one person acted as a bot and the another one as a user;
2. The person, who played bot's role could only ask questions, user could only answer them;
3. Bot asked questions according to the dialogue structure described above.

To get an idea, what information the bot extracts from the interlocutor and how it uses it - each user input is provided with a dataset that system has on the current stage. It includes 7 fields (see Figure 2): problem, object, reasons, experience, circumstances, exception and stage. *Problem* contains problems defined by user. *Object* includes objects affected by these problems. *Reasons* are causes of these issues. *Experience* contains reasons, which user already experienced. Circumstances, where user had these experiences are situated in the *circumstances* field. *Exception* is a found exception from user issue. *Stage* is a step of the Socratic Method, which defines what data should be gathered, and what response user will get.

### 3.3.2 Conversation example.

At the beginning, dataset is empty. Only has a default value in it (see Figure 2).

```
DATA = {'problem': [], 'object': [], 'reasons': [], 'experience': [], 'circumstances': [], 'exception': [], 'stage': 1}
```

Figure 2. The dataset before the start of the dialogue.

Then the chatbot starts the conversation (see Figure 3). Out of user input, the Socratic chatbot extracts 2 main entities: problem and object. The stage is also changes to number 2. After this, the system upgrades dataset.



```
DATA = {'problem': [is sick], 'object': [my dog], 'reasons': [], 'experience': [], 'circumstances': [], 'exception': [], 'stage': 2}
```

Figure 3. Stage 1 - get user statement.

Using data from the user's statement, Socratic asks for user perspective on the problem. With interlocutor's response, bot upgrades current data and gets a new one (see Figure 4).



```
DATA = {'problem': [is sick], 'object': [my dog, she], 'reasons': [sleep a lot, not eat much], 'experience': [], 'circumstances': [], 'exception': [], 'stage': 3}
```

Figure 4. Stage 2 - get reasons.

Then the chatbot tries to draw user past experience by using each described earlier reason separately (see Figure 5).

S So she has never slept a lot?

Only when she is very tired.

```
DATA = {'problem': [is sick], 'object': [my dog, she], 'reasons': [not eat much], 'experience': [sleep a lot], 'circumstances': [very tired], 'exception': [], 'stage': 4}
```

Figure 5. Stage 3 - get past experience.

From the user response bot understands that there are other circumstances with the same "reason" (when the dog is very tired). Therefore, it tries to get an exception, by connecting found circumstances and other reasons (see Figure 6).

S Are you sure that she is not very tired at the moment?

Yes, I am

```
DATA = {'problem': [is sick], 'object': [my dog, she], 'reasons': [not eat much], 'experience': [sleep a lot], 'circumstances': [very tired], 'exception': [], 'stage': 4}
```

Figure 6. Stage 4 - get exception.

However, user response meant that proposed exception was wrong. It signifies that Socratic has to get back to the stage earlier. However, to be sure, to what stage exactly, it controls the available data. In other words, on this stage bot checks received data and removes the unnecessary one. In this dialogue, it became clear that out of 2 proposed reasons at the beginning, only one of them was still valid. So data was updated accordingly to that (see Figure 7).

S I see... This is a complicated problem, so let's repeat all details. So the issue is that your dog is sick, and you assume that because, your dog isn't eating much and is sleeping a lot?

Yes, this is correct

```
DATA = {'problem': [is sick], 'object': [my dog, she], 'reasons': [not eat much], 'experience': [], 'circumstances': [], 'exception': [], 'stage': 2}
```

Figure 7. Stage 5 - control.

After the control stage, the system gets back to the stage 3, because it still has another "reason" (see Figure 8).

S So you don't remember that she has ever not eat much?

Sometimes, when she doesn't like the food

DATA = {'problem': [is sick], 'object': [my dog, she], 'reasons': [], 'experience': [not eat much], 'circumstances': [not like food], 'exception': [], 'stage': 3}

Figure 8. Stage 3 - get past experience.

Received user response meant that there is a chance to get the right exception out of the available data. Therefore, the chatbot attempts to get it, by using new circumstances (see Figure 9).

S Can you say for sure that your dog does not like the food at the moment?

No

DATA = {'problem': [is sick], 'object': [my dog, she], 'reasons': [], 'experience': [not eat much], 'circumstances': [not like food], 'exception': [not like food], 'stage': 4}

Figure 9. Stage 3 - get exception.

To confirm found exception, the chatbot goes to the control stage again (see Figure 10).

S So maybe the problem is that she doesn't like the food and this is the reason, why she isn't eating much?

Yeah, might be so!

DATA = {'problem': [], 'object': [], 'reasons': [], 'experience': [], 'circumstances': [], 'exception': [], 'stage': 5}

Figure 10. Stage 4 - control.

In such a manner, the chatbot found an exception to user statement: the dog might not be sick, but just doesn't like new food. In addition, user decides to finish the conversation (see Figure 11).

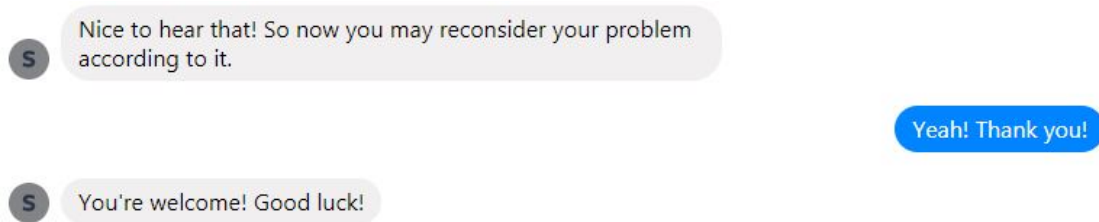


Figure 11. Stage 6 - finish the dialogue.

### 3.3.3 Conversations output.

As already mentioned above, this concrete dialogue and others, made during the design stage, were done to represent the ideal cases of using the Socratic chatbot. Moreover, the main output from them, is that this approach of the Socratic Method worked for the interlocutors. These dialogues made them reconsider their problems and think through the whole situation.

In such a manner, the Socratic chatbot aspires to this conversational model. The next chapter - **Implementation**, describes precisely, how this system is built and how it applies this model.

## 4 Implementation

The implementation of any conversational system may be divided by 2 main stages. The first one is user input analyzing, and the other one is response generating. For the Socratic chatbot, user input analyzing involves text preprocessing, useful information extraction, its modification and saving.

Response generating process consists of method's stage detection, making a response, using predefined patterns and extracted information from user input, and grammar checking. However, in this thesis and current version of the chatbot, user input analyzing is much more complicated than response generating. For that reason, they are combined together. However, before detailed investigation of it, used modules and libraries have to be specified.

### 4.1 Modules and libraries

This project uses multiple Natural Language Processing libraries for functioning. The main one is spaCy [30]. According to the information in the official spaCy repository: "spaCy is a library for advanced Natural Language Processing in Python and Cython." [31] It was chosen for the Socratic chatbot, because it is very fast, provides precise dependency parsing and tokenization, has integrated named entity recognition and continues to actively develop [32].

Along with spaCy, the Socratic chatbot uses textacy for text preprocessing, Natural Language Toolkit (nltk) for sentiment analysis and language-check for the final grammar check.

As written in the official documentation: "textacy is a Python library for performing higher-level natural language processing (NLP) tasks, built on the high-performance spaCy library." [33] Due to its compatibility with spaCy and easy preprocessing approach, it suited the Socratic chatbot. NLTK or Natural Language Toolkit is a "leading platform for building Python programs to work with human language data." [34] With its help the Socratic chatbot does sentiment analysis, which helps in the response generating phase.

Language-check [35] is a "Python wrapper for LanguageTool." LanguageTool [36] is a proofreading service that checks and corrects grammar and style of the text. By using language-check, the chatbot controls and corrects the response for user.



## 4.2 User input analyzing and Response generating

The whole system's high-level architecture may be represented by a flow-chart (see Figure 12):

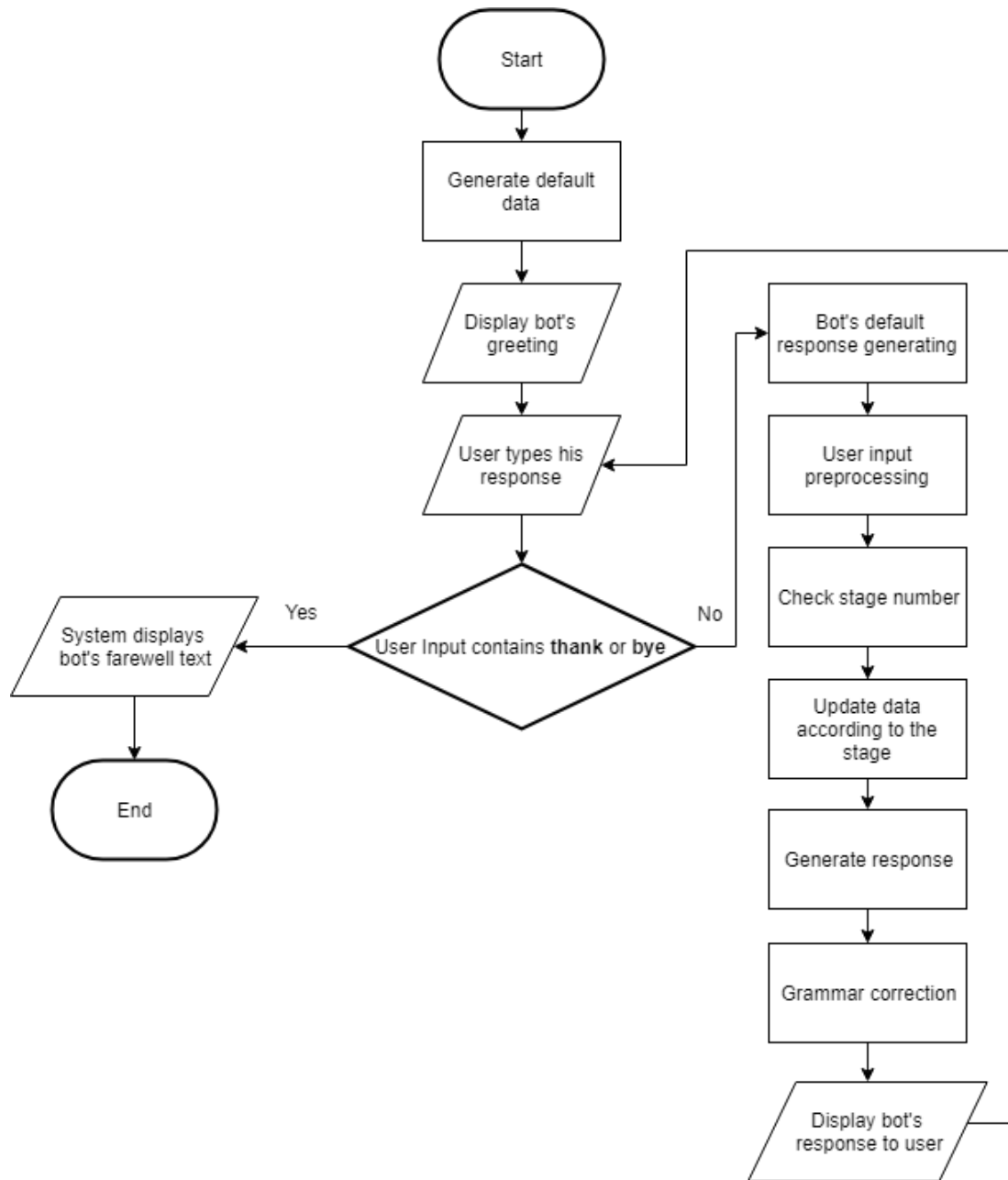


Figure 12. The Socratic chatbot high-level system architecture.

When user starts the chatbot, it automatically generates the default dataset already

described in the **Design** section above. The bot greets user: "Hello, I am a Socratic Chatbot. My goal is to help people better understand their problems. And I'll try my best to help you, my friend! So tell me, what is your problem? What you would like to understand?"

Then user responds to it with his problem, that he would like to discuss. The system checks, does the user reply contains words *thank* or *bye*. If it does, it means the end of conversation, and bot displays the farewell message - "Good luck!" - and finishes the communication.

If conversation continues, then the Socratic chatbot generates default response: "Sorry, I don't understand you. Can you say it again?" This value is used as a backup, so if any error appears, the bot could still sustain the conversation.

After this, user input goes through preprocessing stage. Using tools and libraries described in the previous subsection, the system tokenizes user input, removes punctuation and contraction and applies truecasing. Once it is done, the bot uses dependency parsing to get the root of the sentence. It is necessary for information extraction.

Then system checks stage's number. They are the same as described above in the **Design** section, except that *get user statement* and *get reasons* are combined into 1. However, the system also has two additional stages for numbers 3 (*get exception*) and 4 (*control*). Their goal is to make transitions between steps more user friendly.

As described in the **Design** section, in general, the logic behind every stage is very similar - check, what information is available in the data, extract useful and missing details, and use it in the response generation. The 1. stage (*get user problem and ask for reasons*) is a good example of it.

On this stage, system determines in the first place, whether data contains any *problems*. It is necessary, because every stage may be used multiple times during the conversation (see "Applying the Socratic Method" in the Design section above) and bot's response changes depending on it. If the data doesn't have any *problem*, it means, that before response generating, the chatbot has to extract from user input *problem* and *objects*.

In order to find the problem, Socratic determines the problem root first. The problem root - is a word, from which the Socratic chatbot finds all problem's components. By using dependency tree parsing, system finds all sentence root's children. Then, it searches for the necessary component by its syntactic relation. To understand, for which relation system should look for, every problem statement from the design stage's dialogues was parsed and analyzed. This kind of preliminary analysis was done for every place of the system, where was needed to check syntactic dependency. Consequently, it became clear, that problem root is either a *clausal component* (ccomp) or sentence root itself. For that reason, the chatbot tries to find a ccomp, but if it doesn't find one - the sentence root is considered to be the problem root.

To get the *problem*, the system goes through every child of the problem root and checks its syntactic dependency. If it is not *nominal subject* (nsubj), *marker* (mark) or

*auxiliary* (aux) then it is saved as a *problem*.

*Objects* are extracted the same way, but with different syntactic dependencies. Then *problem* and *objects* are transformed. The transformation consists of converting pronouns into the form suitable for bot's response. E.g., "I" changes to "you". In addition, transformation process includes lemmatization. After this, found *problem* and *objects* are imported into the data. Hereafter, system upgrades the dataset by changing the stage depending on the available information in the data.

When the dataset is updated, the next step is response generating. Every stage contains an individual set of questions (patterns), from where chatbot randomly choose one and adds the necessary data to it. On the first stage, this data is object and problem. For example, if dataset contained *problem* at the beginning of the stage, pattern would be: "So you are saying that [object] [problem]. Why do you think like that?" Otherwise, bot's question is: "I am sorry, but I'm a little bit confused. Can you please repeat, why do you think that [object] [problem]?"

These bot's questions with data placed into them have many grammar mistakes that need to be corrected. For that reason, the next step is grammar correction, which is automatically done by the language-check library. Then response is displayed to the user and bot waits again for his response.

This logic works for every stage, but with small changes according to its number. On the 2. stage (*get past experience*), the system searches not for *problems*, but for *reasons*, on the 3. (*get exception*) for *circumstances* and so on.

The only change happens on the stage number 4 (*control*) and 41, where bot's response depends up not on the available data, but on the text sentiment. For that reason on these stages system uses sentiment analysis to determine, whether user input was affirmative or negative, and chooses response pattern based on it.

To see, how well this system completed its tasks, it was tested multiple times. It is described in the next chapter - **Validation**.

## 5 Validation

Validation is the essential step of any development. It helps to check current solution from different perspectives and make product better. The validation process of the Socratic chatbot consisted of 2 iterations. Iteration included 3 steps:

1. The chatbot was given to testers. They communicated with it, and their conversations were stored;
2. Testers gave their feedback on the solution and the idea of the bot;
3. Chatbot was improved according to their feedback and interactions.

### 5.1 First iteration

#### 5.1.1 Experiment setup

The first iteration was done with 3 testers. All of them had different backgrounds: software engineering, biology and computer science. None of them was aware of the Socratic Method, but they have communicated with chatbots before. However, their interactions were limited by asking about the weather, suggesting restaurants, etc.

Testers were told about the Socratic Method and about the purpose of this chatbot. After that, testers were asked to communicate with the bot. Also, were defined next rules for dialogues:

1. User can only answer to questions;
2. Answers should be direct and clear;
3. User responses might be maximum 2-3 sentences.

During conversations, supervisor helped testers to formulate responses to make them more understandable to the bot. Were created multiple dialogues, but they have all been very similar in case of technical issues. One of these conversations can be seen below (Figure 13).

#### 5.1.2 Feedback

It was clear that bot has many problems. The main ones are data extraction, grammar, questions and short affirmative responses.

The feedback from testers was taken in a free form. They were positive about the idea of this chatbot, but at the same time, due to poor implementation, they could not experience any good outcome from these conversations. A lot of grammar mistakes and inability to sustain a logical conversation flow, made it impossible for testers to understand bot's responses and rethink problems defined in the dialogues.

```

Bot: Hello, I am a Socratic Chatbot. My goal is to help people better understand their problems. And I'll try my best to help you, my friend! So tell me, what is your problem? What you would like to understand?
User: Hello. I think that I have heart problems.
Bot: Okay, so you assume that you have heart problem. What made you think like that?
User: Sometimes, when I'm stressed my heartbeats become more frequent and I feel pain in my left lung.
Bot: Can you remember, did you ever sometimes when you be stress become more frequent?
User: Yes, I have.
Bot: Let me clear it one more time. There was no experience, when your heartbeat sometimes when you be stress become more frequent?
User: No, there was an experience like that
Bot: Can you say for sure that you not like an experience at the moment?
User: Yes, I can
Bot: Sorry, I don't understand you. Can you say it again?
User: Yes, I can say it for sure
Bot: Can you remember, did your heartbeat ever feel in pain?
User: No
Bot: Are you sure that your heartbeat not like an experience at the moment?
User: Yes
Bot: Sorry, I don't understand you. Can you say it again?
User: I said yes
Bot: I apologize, but I entangled myself in this discussion. Can you please explain one more time, why do you assume that you and your heartbeat have heart problem?
User: Bye.

```

Figure 13. Dialogue from the first iteration.

### 5.1.3 Changes

In order to improve the chatbot, testers' dialogues were debugged sentence by sentence to find and correct issues. List of changes during this iteration:

1. The data extraction algorithm was updated. Before system checked and added to data only children, which were closest to the root. After this update, system walks through every acceptable child. Also, were updated applicable syntactic relations, and words order of the extracted information is now the same as in the user input.
2. The lemmatization of the extracted information was removed to prevent some grammar mistakes. Grammar correction module didn't need it, and also bot's lemmatized response was harder to understand then the one without it.
3. Questions were reconsidered and updated, so that users' answers to them could be interpreted in only one way possible. Also after the update, extracted information could be placed into bot's responses more accurately. For example, on the 2. stage (get past experience), bot could response with: "So [object] has never [problem]?"; "Can you remember, did [object] ever [problem]?". In this case, user answer "yes", meant different things. As a response for the first question, it indicated that [object] has no experience of the [problem]. However, as a response for the second question, it indicated exactly the opposite. To eliminate these inaccuracies, these questions were changed to: "Has [object] ever experienced, that [problem]?", "Do you remember, that [object] has ever experienced, that [problem]?".
4. To process shirt responses during the second stage (get past experience), the new stage was added. Its goal is to process affirmative or negative responses from user about his past experiences. With this added stage, the dialogue flow became much

more user-friendly, and now when interlocutor will answers monosyllabically, bot will accept it and ask questions based on it.

After these changes, the chatbot became capable of sustaining these conversations. For example, bot was able to extract information so, that it is understandable for user (see Figure 14). To test this solution again on different people and under near-natural conditions, the 2. iteration was done.

```
Bot: Hello, I am a Socratic Chatbot. My goal is to help people better understand their problems. And I'll try my best to help you, my friend! So tell me, what is your problem? What you would like to understand?
User: Hello. I think that I have heart problems.
Bot: Okay, so you assume that you have heart problems. What made you think like that?
User: Sometimes, when I'm stressed my heartbeats become more frequent and I feel pain in my left lung.
Bot: Have you ever experienced, that your heartbeats become more frequent?
User: Yes
Bot: Please, describe your experience.
```

Figure 14. Fixed reason extraction.

## 5.2 Second iteration

### 5.2.1 Experiment setup

The second iteration was done with other 7 testers. Their backgrounds were different as well: 2 - computer science, 2 - law, 1 - law and political science, 1 - software engineering and 1 - biology. This time, 2 testers were aware of the Socratic Method and used its modern approach in real life to teach other people new concepts. However, the experience of using chatbots was the same - only ordering or buying something through them.

This time testers were also briefed about the used Socratic Method approach, goal of this bot and conversation rules. After this, they started to communicate with the chatbot.

However, the main difference between 1. and 2. iterations was that the supervisor didn't help testers to formulate responses or understand bot's questions. In other words, these conditions were really close to real: full freedom of conversation and only 2 participate in the dialogue - user and bot. And again, most of the dialogues were similar in terms of technical issues. See one of these conversations on Figure 15.

### 5.2.2 Feedback

This time feedback was gathered through the questionnaire. It was made in such a way as to validate current solution - generalist and rule-based chatbot that uses classic Socratic Method - from user perspective. The questionnaire included 7 questions.

The first question was about the background of testers. As already mentioned above 2 users had a background in computer science, 2 in law, and 1 in biology, 1 in software engineering and 1 in law and political science.

Bot: Hello, I am a Socratic Chatbot. My goal is to help people better understand their problems. And I'll try my best to help you, my friend! So tell me, what is your problem? What you would like to understand?  
 User: I get in very aggressive disagreements with my coworkers at work.  
 Bot: Okay, so you assume that you get disagreements in. What made you think like that?  
 User: I feel very tense and upset when there are disagreements  
 Bot: Do you remember, that you have ever experienced, that you feel very tense and upset when there are disagreements?  
 User: Yes  
 Bot: Please, describe your experience.  
 User: Sometimes we discuss about different ideas for projects and ways to do the activities, but everyone has different ideas  
 Bot: Are you sure that you not ideas about to activities do at the moment?  
 User: Yes  
 Bot: Sorry, I don't understand you. Can you say it again?  
 User: Yes  
 Bot: I am sorry, but I'm a little confused. Can you please repeat, why do you think that you get disagreements in?  
 User: Everyone is very outspoken about their ideas and act very aggressively  
 Bot: Have you ever experienced, that everyone is very outspoken about their ideas and act very aggressively?  
 User: Yes  
 Bot: Please, describe your experience.  
 User: Even with topics like football differences lead to conflicting arguments  
 Bot: Can you say for sure that you not ideas about to activities do and arguments to at the moment?  
 User: Yes  
 Bot: Sorry, I don't understand you. Can you say it again?  
 User: I said - yes!  
 Bot: Sorry, I lost the thread of conversation. Let's start again from your assumptions. Why did you assume that you get disagreements in?  
 User: Sorry, but this conversation is going nowhere. Bye!  
 Bot: Good luck!

Figure 15. Dialogue from the 2. iteration about work.

In the second question (see Figure 16), testers chose about what topic, they were communicating with the bot. 4 of them were talking about academic topic, 2 about personal relationships and 1 about health.

### What topic was discussed with the chatbot?

7 responses

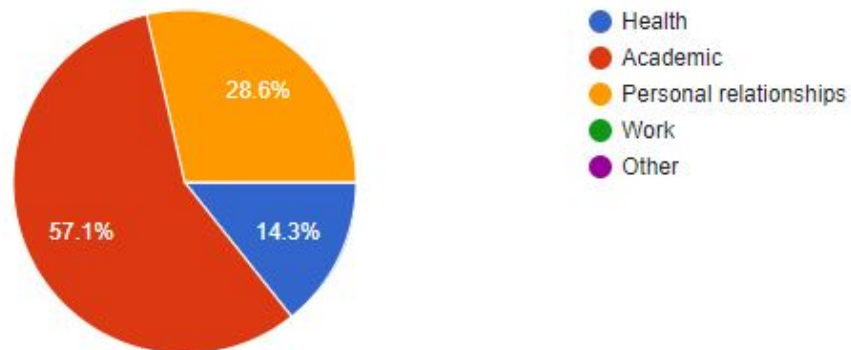


Figure 16. Topics for dialogues.

The third questions was about rethinking user's problems (see Figure 17). This was one of the main bot's goals, and results showed that 2 testers reconsidered their problem and other 2 weren't sure.

## Did you rethink your problem by interacting with the bot?

7 responses

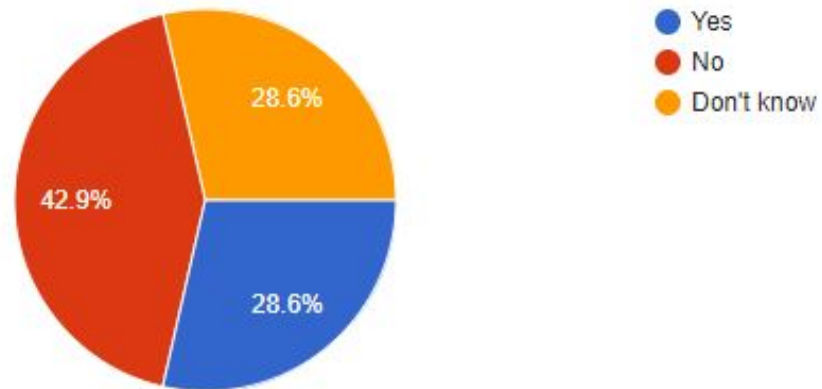


Figure 17. How well bot made testers rethink their problems.

The next question checked, how understandable were bot's questions (see Figure 18). All 7 testers answered that chatbot responses were clear, but for 4 of them, replies weren't always clear.

## Was the replies by the bot understandable?

7 responses

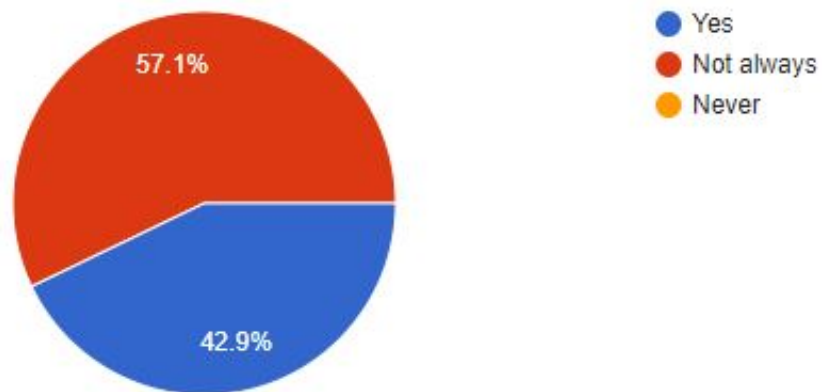


Figure 18. How well bot formulated its responses.



The fifth question showed that for 2 testers, the bot was not able to sustain a logical conversation flow, for 1 tester it happened only sometimes and for 1 it always worked (see Figure 19).

### Did the bot sustain a logical conversation flow?

7 responses

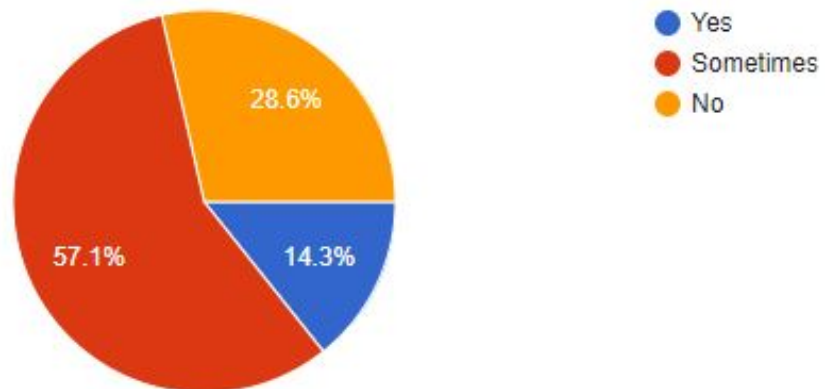


Figure 19. How well bot sustained a logical conversation flow.

The next question showed that for 3 people their dialogues ended suddenly. However, the other 4 testers answered positively (see Figure 20).

### Did you reach the end of the dialogue?

7 responses

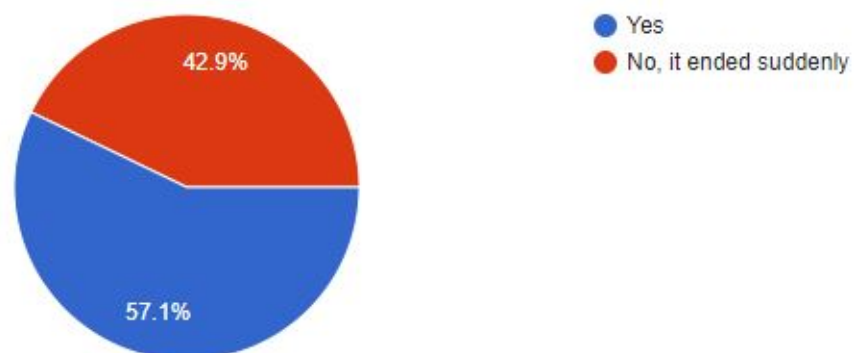


Figure 20. How many testers reached the end of the dialogue.

The last question was about the ability of the bot to help in finding possible solution for user's problem (see Figure 21). Out of 7 testers, only 1 answered positively, 2 were not sure and 4 people answered negatively.

## Did it help you to get an idea of possible solution?

7 responses

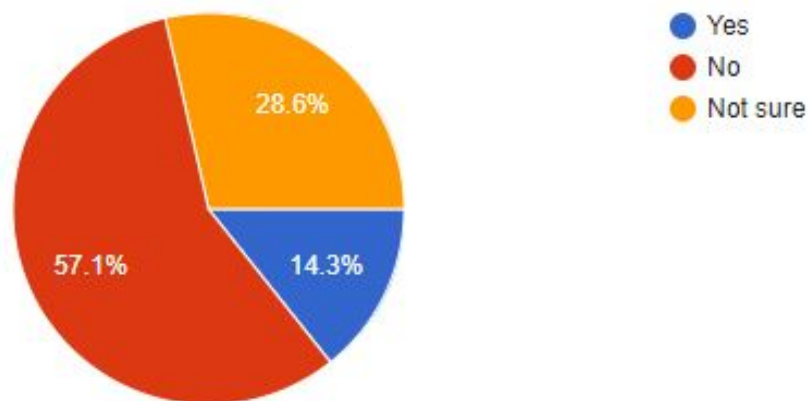


Figure 21. Testers feedback about finding possible solution.

### 5.2.3 Results

The feedback shows that this bot can work in some cases and really help users to rethink and get an idea of possible problem's solution. Although, it happens very seldom. The second iteration proved that, when the user doesn't have any limits in communication, it is very hard for the bot to sustain logical conversation flow, make replies understandable, and therefore, help users to rethink and find problems' solutions.

It is clearly seen in the dialogue (see Figure 15), when up to the *get experience* stage, the chatbot was able to reply adequately and extract right information. However, the system was not able to "understand" the big and hard sentence on this stage: "Sometimes we discuss about different ideas for projects and ways to do the activities, but everyone has different ideas". Therefore, it led to the wrong information extraction and consequently to incomprehensible response: "Are you sure that you not ideas about to activities do at the moment?"

In addition, this bot's reply shows the other problem, which causes illogical conversation flow: pre-defined questions. The chatbot tries to find an exception by applying user's past experience to the current moment. However, in some dialogues, testers talked about general problems, which don't have to happen "at the moment". Moreover, it

cannot be solved by just pattern matching words like "sometimes" or "now and then", because people don't always use them to show that something happens in general and not in concrete time.

These tests made it clear, that the Socratic chatbot, which is generalist and rule-based bot, can work as it supposed to, but only in some cases. Most of the time it fails to get the right information, because there are always exceptions to the rules, and scripted system that has a clear goal (find an exception to the user's statement) is unable to handle them. However, the fact that it helped some users to rethink and better understand their problem, proves that the chatbot, which directly uses the Socratic Method, can be built and work successfully.

### **5.3 Future development**

In order to make this system better with the current approach, will require more dialogues (use cases) to create more rules. It will especially help with data extraction. Some grammar rules can be defined manually as well, if there is no better automatic solution.

Also should be implemented functionality to handle multi-level problems. E.g. in the dialogue about heart problems (see Figure 13), the problem in the first user response sounds like: "I have heart problems". Although, the second user input can be also considered as a problem, but inside the first one: "When I'm stressed my heartbeats become more frequent and I feel pain in my left lung". It can be done by using more complicated database system.

Finally, to make the Socratic chatbot more user friendly, it should have more diverse response patterns.

No changes will remove the rules' limitations, but it will definitely improve the user experience and the number of cases, where it works correctly, will be increased.

## 6 Conclusion

The goal of this thesis was to explore conversational systems, the Socratic questioning method and try to build a chatbot that would use this method.

During this work, were examined chatbots in general, from their classification and implementation process to their role in society. Some exemplars such as Woebot, X2 AI, ELIZA and Elizabeth were explored in more depth.

The Socratic Method, its algorithm, different approaches and use cases were examined in details. Its classic approach was used in several conversations.

By combining gained knowledge about conversational systems and the Socratic Method, was implemented and tested the Socratic chatbot. Its goal was to use the classic Socratic Method to make people rethink and better understand their problems. The Socratic chatbot [12] was built according to the multiple real dialogues and by using Natural Language Processing techniques and tools such as dependency tree parsing, named entity recognition, sentiment analysis and automatic grammar correction.

To validate the solution, was done 2 iterations, which included testing, gathering feedback and bot's upgrading according to it. Overall, 10 people with different backgrounds tested the Socratic chatbot. During testing, the system could sustain conversations using the Socratic Method and in some cases made users to rethink their concerns. However, in most instances, bot failed to extract the right information from the user input, use correct grammar and sustain a logical conversation flow.

In the end of the 2. iteration current solution was analyzed, considering users' feedback. Was drawn a conclusion, that the Socratic chatbot works correctly only in some cases, because as a generalist and rule-based bot it cannot handle appearing in the user input exceptions to these rules.

Although, despite this inference, system possible future improvements were also described. Further development plans include creation of more rules, implementing multi-level problem functionality and more diverse response patterns. It will require more dialogues (use cases) for analysis.

Judging by the result and testers' feedback, the Socratic chatbot [12] fulfilled its goals and has a potential for the future development.

## References

- [1] Rick Garlikov. Teaching by Asking instead of Telling. [http://www.garlikov.com/Soc\\_Meth.html](http://www.garlikov.com/Soc_Meth.html). (14.05.2018).
- [2] Jeffrey G. Rutter, Robert D. Friedberg. Guidelines for the effective use of Socratic dialogue in cognitive therapy. *Innovations in clinical practice*, 1999, 481–490.
- [3] Joseph Weizenbaum. ELIZA A Computer Program For the Study of Natural Language Communication Between Man And Machine. *Communications of the ACM*, 1966, nr 1, 36–45.
- [4] Woebot official website. <https://woebot.io/>. (14.05.2018).
- [5] Siri official website. <https://www.apple.com/ios/siri/>. (14.05.2018).
- [6] Amazon Alexa official website. <https://developer.amazon.com/alexa>. (14.05.2018).
- [7] GoogleAssistant official website. <https://assistant.google.com/>. (14.05.2018).
- [8] Messaging apps are now bigger than social networks. <http://www.businessinsider.com/the-messaging-app-report-2015-11>. (14.05.2018).
- [9] Johnson K. Facebook Messenger hits 100,000 bots. *Venture Beat*, <https://venturebeat.com/2017/04/18/facebook-messenger-hits-100000-bots/>. (14.05.2018).
- [10] 80% of businesses want chatbots by 2020. <http://www.businessinsider.com/80-of-businesses-want-chatbots-by-2020-2016-12>. (14.05.2018).
- [11] Foye L. Chatbot conversations to deliver \$8 billion in cost savings by 2022. *Analyst Xpress*, <https://www.juniperresearch.com/analystxpress/july-2017/chatbot-conversations-to-deliver-8bn-cost-saving>. (14.05.2018).
- [12] The Socratic chatbot - official github repository. <https://github.com/janmoppel/socratic-chatbot>. (14.05.2018).
- [13] Gregory Vlastos. The socratic elenchus. *Oxford Studies in Ancient Philosophy. Volume 1.*, Oxford University Press, 1983, vol. I, 27–58.
- [14] Maxwell M., Melete. How To Use the Socratic Method. [http://www.socraticmethod.net/how\\_to\\_use\\_the\\_socratic\\_method/using\\_the\\_socratic\\_method.html](http://www.socraticmethod.net/how_to_use_the_socratic_method/using_the_socratic_method.html). (14.05.2018).

- [15] Maxwell M. The Moral Bankruptcy of Faith. <http://www.socraticmethod.net/morality/page1.htm>. (14.05.2018).
- [16] Padesky C. A. Socratic Questioning: Changing Minds or Guiding Discovery? <https://padesky.com/newpad/wp-content/uploads/2012/11/socquest.pdf>. (14.05.2018).
- [17] Plato. Meno. <https://www.staff.ncl.ac.uk/joel.wallenberg/ContextsJoelGeoff/meno.pdf>. (14.05.2018).
- [18] Chatbot definition. <https://en.oxforddictionaries.com/definition/chatbot>. (14.05.2018).
- [19] Types of Bots: An Overview. <http://botnerds.com/types-of-bots/>. (14.05.2018).
- [20] I'm Home Routine. <https://assistant.google.com/services/a/uid/0000001b6a2ed272>. (14.05.2018).
- [21] Rahul. Say Hello To Uber On Messenger. <https://www.uber.com/newsroom/messengerlaunch/>. (14.05.2018).
- [22] CNN bot. <https://botlist.co/bots/cnn>. (14.05.2018).
- [23] Dialogflow official website. <https://dialogflow.com/>. (14.05.2018).
- [24] Wit.ai official website. <https://wit.ai/>. (14.05.2018).
- [25] Cortana official website. <https://www.microsoft.com/en-us/cortana>. (14.05.2018).
- [26] X2 AI - Mental Health chatbot. <http://x2ai.com>. (14.05.2018).
- [27] Fitzpatrick KK, Darcy A, Vierhile M. Delivering Cognitive Behavior Therapy to Young Adults With Symptoms of Depression and Anxiety Using a Fully Automated Conversational Agent (Woebot): A Randomized Controlled Trial. *JMIR Ment Health* <https://mental.jmir.org/2017/2/e19/>. (14.05.2018).
- [28] Shawar B.A., Atwell E. A Comparison between Alica and Elizabeth. *White Rose Research Online* <http://eprints.whiterose.ac.uk/81930/>. (14.05.2018).
- [29] Masayuki Okamoto, Yeonsoo Yang, and Toru Ishida. Wizard of oz method for learning dialog agents. In Matthias Klusch and Franco Zambonelli, editors, *Cooperative Information Agents V*, page 22, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

- [30] spaCy official website. <https://spacy.io/>. (14.05.2018).
- [31] spaCy official github repository. <https://github.com/explosion/spaCy>. (14.05.2018).
- [32] spaCy Facts and Figures. <https://spacy.io/usage/facts-figures>. (14.05.2018).
- [33] textacy: higher-level NLP built on spaCy. <https://textacy.readthedocs.io/en/stable/>. (14.05.2018).
- [34] Natural Language Toolkit official website. <https://www.nltk.org/>. (14.05.2018).
- [35] language-check official page. <https://pypi.org/project/language-check/>. (14.05.2018).
- [36] Language Tool official website. <https://languagetool.org/>. (14.05.2018).

# **I. Licence**

## **Non-exclusive licence to reproduce thesis and make thesis public**

**I, Jan Moppel,**

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:
  - 1.1 reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and
  - 1.2 make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

of my thesis

### **Socratic chatbot**

supervised by Taivo Pungas and Sven Aller

2. I am aware of the fact that the author retains these rights.
3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, 14.05.2018