2022

ZAKLJUČNA NALOGA (FINAL PROJECT PAPER)

PELIVANOV

UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN INFORMACIJSKE TEHNOLOGIJE

ZAKLJUČNA NALOGA
(FINAL PROJECT PAPER)

AVTOMATIZIRAN SISTEM ZA POMOČ ŠTUDENTOM S SPLETNIM KLEPETALNIM ROBOTOM
(AUTOMATED STUDENT SUPPORT SYSTEM WITH ON-LINE CHATBOT)

DIMITAR PELIVANOV

UNIVERZA NA PRIMORSKEM
FAKULTETA ZA MATEMATIKO, NARAVOSLOVJE IN INFORMACIJSKE TEHNOLOGIJE

Zaključna naloga
(Final project paper)

Avtomatiziran sistem za pomoč študentom s spletnim klepetalnim robotom

(Automated student support system with on-line chatbot)

Ime in priimek: Dimitar Pelivanov
Študijski program: Računalništvo in informatika
Mentor: izr. prof. dr. Klen Čopič Pucihar
Somentor: asist. dr. Domen Šoberl

Koper, **avgust 2022**

Ključna dokumentacijska informacija

Ime in PRIIMEK: Dimitar PELIVANOV

Naslov zaključne naloge:

Avtomatiziran sistem za pomoč študentom s spletnim klepetalnim robotom

Kraj: Koper

Leto: 2022

Število listov: 38 Število slik: 13

Število referenc: 16

Mentor: izr. prof. dr. Klen Čopič Pucihar

Somentor: asist. dr. Domen Šoberl

Ključne besede: avtomatiziran sistem za pomoč študentom, podpora študenta, chatbot

Izvleček:
Glavna tema diplomske naloge je implementacija sistema za pomoč študentom, specializiranega za pomoč pri odgovarjanju na vprašanja v zvezi s študentskimi zadevami na Fakulteti za matematiko, naravoslovje in informacijske tehnologije v Kopru. Najprej govorimo o tem, kako deluje chatbot, opišemo njegovo arhitekturo, nato pa je obravnavamo raziskavo razpoložljivih komercialnih in brezplačnih orodij, skupaj z njihovimi prednostmi in slabostmi za naš cilj. Predstavljeni so vsi potrebni koraki za implementacijo takega chatbota, vključno s pisanjem spletne aplikacije za klepet med strežnikom in odjemalcem in nekaj izvedljivih algoritmov strojnega učenja za pridobivanje pomena iz prejetih vprašanj. Zaključimo s predstavitvijo rezultatov in pregledom možnih prihodnjih izboljšav.

Key document information

Name and SURNAME: Dimitar PELIVANOV

Title of the final project paper:

Automated student support system with on-line chatbot

Place: Koper

Year: 2022

Number of pages: 38 Number of figures: 13

Number of references: 16

Mentor: Assoc. Prof. Klen Čopič Pucihar, PhD

Co-Mentor: Assist. Domen Šoberl, PhD

Keywords: automated student support service, student support, chatbot

Abstract:
This thesis main topic is the custom implementation of a student support system, specialized for helping answering the questions regarding student affairs at the Faculty of Mathematics, Natural Sciences and Information Technologies in Koper. First is explained how chatbot works, their architecture and than a research of the available commercial and free tools is discussed together with their advantages and disadvantages for our particular goal. Next we present all the necessary steps to implement such a chatbot from scratch, including writing a custom server-client chat application and some viable machine learning algorithms for extracting the meaning from the received questions. We finish by presenting the results and going over potential future enhancements.

Acknowledgments

I take this opportunity to express my tremendous gratitude to University of Primorska and the Faculty of Mathematics, Natural Sciences and Information Technologies for giving me the opportunity to pursue my passion in information technologies. The knowledge that I have acquired during my studies will without a doubt be a great stepping stone in my future endeavors.

I would like to express my gratitude to the people who supported my efforts in writing this thesis. First, my mentor, izr. prof. dr. Klen Čopič Pucihar, whose knowledge and insight into the subject matter steered me until the completion of the thesis. Special thanks to my comentor, dr. Domen Šoberl, who provided invaluable feedback and meticulous editing on my analysis. Lastly, I am incredibly grateful for the unwavering support and love from my family and friends.

Dimitar Pelivanov
Avgust, 2022

Contents

1 Introduction **1**

2 How chatbots work **3**
2.1 Chatbot Categorization 3
2.1.1 Rule-based chatbots 4
2.1.2 AI Chatbots 4
2.1.3 Hybrid Chatbots 4
2.2 Chatbot Architecture 5
2.2.1 User Interface 5
2.2.2 User Message Analysis 5
2.2.3 Dialog Management 7
2.2.4 Backend 7
2.2.5 Response Generation 8

3 Commercial chatbot tools **9**
3.1 Microsoft's Bot Framework 9
3.2 Dialogflow 9
3.3 Botsify 10
3.4 Botpress 11
3.5 IBM Watson Assistant 11
3.6 Conclusions 11

4 Implementation **13**
4.1 Requirements Analysis 13
4.1.1 Functional requirements 13
4.1.2 Non-Functional requirements 14
4.2 System architecture 15
4.2.1 Client-server model 15
4.2.2 Deployment diagram 16
4.2.3 Technologies used 17
4.3 Chatbot implementation 17
4.3.1 BERT and the transformer 18
4.3.2 The training dataset 19
4.3.3 Universal sentence encoder model 19
4.3.4 Final improvements 20
4.4 Results 22

4.4.1 Sample scenarios 22

5 Conclusion **25**

6 Povzetek naloge v slovenskem jeziku **26**

7 Bibliography **28**

List of Figures

1 Use Case Diagram 14
2 Schema of our system architecture 15
3 Deployment Diagram 16
4 Masked Language Model 19
5 Semantic Similarity 20
6 Classification 20
7 Bert model example 21
8 Universal Sentence Encoder example 21
9 Conversation flow 22
10 Interaction between the chatbot and the user. 23
11 Interaction between the chatbot and the user. 23
12 Establishing the connection with an operator. 24
13 Communication between the user and the operator 24

List of Abbreviations

AI	Artificial Intelligence
ML	Machine Learning
NLP	Natural Language Processing
NLU	Natural Language Understanding
NLG	Natural Language Generation
API	Application Programming Interface
FAQ	Frequently Asked Questions
MBF	Microsoft Bot Framework
CRM	Customer Relationship Management
UI	User Interface
BERT	Bidirectional Encoder Representations from Transformers
HTTP	Hypertext Transfer Protocol
MLM	Masked Language Modeling
VUI	Voice User Interface

1 Introduction

Technology today plays enormous role in human life and improves from day to day. Every one of us almost every day must interact somehow with some kind of technology for gathering some kind of information. Every interaction between the computer and human takes one of the most important place in technological product. Artificial Intelligence is making automation available in every field of the technology. By developing and analyzing sophisticated software and hardware, referred to as intelligent agents, that are capable of carrying out a variety of tasks, artificial intelligence [13] has had an impact on how we engage in our daily lives.

There are different approaches that offer replacements for the time-consuming search for information. One of the turning points are the natural language user interfaces visible today in chatbots [6]. Chatbots are about simulation of user's dialogue. Depending on users behavior they are able to provide users with the information they need. Bots that are using artificial intelligence have certain advantages in comparison to the bots with preset responses, most notably an ability to adapt to users interaction and improve their communication skills in time. They have become a very popular technology for interaction between human and computer, especially due to the increased use of mobile platforms, where texting is often more comfortable than interacting with a graphical user interface. From the standpoint of human psychology, conversation may feel more natural, so a conversation-based user interface may soften the dividing line between a human and a machine type of support.

Popularity of the chatbots increases all the time especially in business these days. Businesses are using them as leverage for improvement of their customers experience. Chatbots are beneficial in a number of other industries, including education [10], business and e-commerce [15], health [14], and entertainment [8], in addition to mimicking human interaction and amusing users [2]. Although there are other reasons why people use chatbots, including amusement, social factors, and unique interactions, productivity is the main driver. Additionally, increased use of chatbots in business is due to the fact that they lower service costs and can serve multiple users at once. Users find chatbots to be friendlier and more appealing than the static content [11]. Chatbots are ubiquitous, especially in the realm of marketing, with so much of the globe trending toward automation. They can also help businesses increase conversions during marketing campaigns [16].

Most of the time, users perceive chatbots as nice friends rather than merely helpful aides. Another crucial component to humanizing a chatbot is emotion, and there are numerous methods for creating one. Part of user requests are emotional. The advancement in sentiment analysis and machine learning [7] gave chatbots the ability to respond emotionally to users [4]. A chatbot behavior, appearance, and other aspects linked to its manufacturer, privacy concerns, and protection all affect how much trust a user will place in it as a result of their use. AI advancements improve chatbots ability to simulate human beings in dialogue. But when compared to human-human conversation, there are observable differences in the content and quality of human-chatbot contact.

The main question of this research is how to make the educational experience even better for students at our faculty. I attempt to approach this matter by implementing a student support system, specialized for helping answering the questions regarding student affairs at The Faculty of Mathematics, Natural Sciences and Information Technologies in Koper.

In this thesis I will first research the existing commercial and free tools for building chatbots, their features and limitations for our specific purpose. Then I will consider a custom implementation of a support system using the JavaScript language with Node.js environment. I will present all the necessary steps to implement such a chatbot from scratch, including writing a custom server-client chat protocol, text parsing routines, and some viable machine learning algorithms for extracting the meaning from the received questions. I will start with the simplest approach of if-then rules and continue with more elaborate prediction models. In cases when the algorithm fails to interpret the question, the question will be forwarded to a live person.

2 How chatbots work

A chatbot, sometimes known as a bot, is basically a computer system that can carry out automated conversations. Bots may also be used in messaging systems. While chatbots are comparable to standard messaging apps, they differ in that one of the message recipients is a machine. In other words, the circumstance is similar to when a person is chatting with a computer. This chatbot could appear to be a typical messaging program. The user interface serves as the point of contact with the user. Although chatbots are simple to use, they are sophisticated in their background goals. Most chatbots save conversation logs, which are used by the developer to comprehend user demands. The chatbot interaction is then improved using the conversational logs. Conversation logs give administrators access to a complete list of end-user conversations with bot instances. The conversation log contains all utterances sent to the bot and all responses returned. Conversation logs can be used to analyze user and bot behavior. For example, someone can check the conversation to see if the end user is interacting with the bot as expected. Moreover, the chatbot can be trained through the examination of tens of thousands of chat logs from real people.

To begin with, the user needs a computer or a handheld device to access a chatbot's user interface. The chatbot user interface will provide a text field where the user can enter text. Furthermore, the user-supplied content that is entered in a sentence will then be *chunked*. Chunking [1] in this context refers to the process of breaking up the text into individual words. Several of these are than recognized as significant and tagged. They are later being used in the matching process, where they serve as keywords. The chatbot then compares the keywords that emerged from the chunking procedure with a pattern. If a match is found the chatbot outputs the pre-programmed or learned response in the case of AI based chatbot.

Chatbots come in a variety of forms, and each one operates differently. The question-and-answer type of chatbots are the easiest to set up. They are mostly knowledge-based or rule-based, and they are only capable of responding to a predetermined set of queries, set of rules. Alternatively, chatbots that fully utilize AI and ML capabilities can resemble human communication and improve user experience.

2.1 Chatbot Categorization

We may categorize chatbots into three groups based on whether they operate using artificial intelligence and machine learning, a set of predetermined set of rules or both combined.

2.1.1 Rule-based chatbots

Currently, this is the most basic type of chatbots. The conversational flow of the chatbot is governed by predefined rules. Because they use a straightforward true-false algorithm to comprehend user inquiries and deliver relevant answers, rule-based chatbots are simpler to develop. Although the underlying technology may be less complex, this strategy emphasizes the importance of knowing the target audience. Whatever bot technology is used, it is important to understand the needs and habits of the users. Because they are pre-programmed, it is simpler to regulate the responses they produce. They do, however, have significant limitations or weaknesses. Users of rule-based chatbots are limited by a predetermined range of options. The main distinction is that because the dialogue is built on a decision tree that always works, they won't be in a position where the chatbot will not be able to comprehend user inquiries. Because typing might be tedious, a list of predefined options is frequently preferable. Additionally, the conversation always moves toward a target that is specified.

However, these chatbots are not a good choice for more complex scenarios where diverse interactions need to be considered.

2.1.2 AI Chatbots

AI chatbots simulate human-like conversations by utilizing large sets of data. They should improve as they gather new information and learn from it as the amount of available data increases. AI-powered chatbots may interpret natural language, but they also follow a predetermined path to ensure that user concerns are resolved. They are able to recall both, the user choices and the conversation context. These chatbots can change points of conversation as needed and respond to arbitrary user requests at any time. To comprehend people, these chatbots employ machine learning, artificial intelligence, and natural language processing. Natural language processing is the ability of a computer to comprehend and analyze human speech, identify the appropriate response, and respond in a manner that is understandable to a human. The aim of NLP is to make computer and human connection feel like two people talking to each other. AI chatbot setup is time-consuming. These chatbots need to be trained, which requires a lot of data and in the case of customers service, this data must be specialized. Such a project requires a significant time commitment. Although we might assume that AI chatbots would optimize themselves, this is rarely the case. Developing an AI chatbot requires a continuous work.

2.1.3 Hybrid Chatbots

Hybrid chatbots use the benefits of both AI-based and the rule-based chatbots. In order to respond to user questions in a specified way, hybrid chatbots are trained to say certain things based on their predefined rules. Hybrid chatbots can comprehend purpose and context and also do some rule-based activities. However, they can also employ natural language processing to determine the user's purpose. They became a well-rounded tool for firms to communicate with customers as a result [3]. Hybrid chatbots overcome some limitations of the rule-based chatbots, but when the complexity of the bot grows, maintaining the rules can be challenging.

2.2 Chatbot Architecture

A chatbot is a piece of software that facilitates spoken or written conversation with people to assist them in completing a task. The architecture of a chatbot is quite similar to a typical client-server web application. The main difference is that while a typical client-server web application works with structured data, a chatbot works with unstructured data.

To the user it appears as a program that replies to his questions. Under the hood, the messaging interface sends raw data to the client-server, the middle layer in the architecture analyzes the text and produces insights, then finally the backend API is called to carry out the desired activity. The architecture carries out the process of comprehending the input and creating a response, or finding a proper predefined response.

Let us now discuss these components in more detail.

2.2.1 User Interface

User engagement with a chatbot occurs through front-end platforms. The chatbot is activated when it gets the user's request through a text-based application. These are platforms that communicate directly with users, such as websites or mobile applications. Among the most known are Facebook Messenger, Slack, etc.

Chatbots are an example of a user interface that uses natural language to interact with a human. This means that users can communicate with a computer as with another human being. Beside the text based user interfaces, there are also other options on the market, very popular being based on voice interaction or voice user interfaces (VUI). They allow users to interact through speech. The most known such devices are Alexa, Siri and Google Assistant. The type of interaction is especially close to the human type of communication.

2.2.2 User Message Analysis

The user interface sends the user's input to the message analyzer, which determines the user's intention and extracts entities employing machine learning or pattern matching techniques. The user's message can be handled through natural language processing or kept as plain text, which preserves all of the syntactical and grammatical structures of the input.

To determine an appropriate response, the chatbot must first identify the user's intention. Different user inputs are mapped to different intentions. Intentions may also contain parameters known as entities, which provide additional information about an intent. A spell checker that is responsible for correcting user's spelling errors may also be included. This can lead to better intent classification. When considering multilingual chatbot users, an automated translation component can also be included. This way the user language is identified, and if needed, it is translated to the language of the chatbot. All these methods fall into the category of Natural Language Processing.

Natural Language Processing

The idea of natural language processing (NLP) is to achieve human level language processing by computer systems. This research area is divided into two subfields, the natural language understanding (NLU) and natural language generation (NLG), which aim to perform the opposite tasks. The NLG is all about how the structured data, generated by the machine, gets converted into text that can be read by people, while NLU is focused on identifying patterns in unstructured input to interpret the meaning of human writing. NLU uses two methods, one is called the intent classifier or intent classification method and the other entity extractor.

Intent classification [9] is a process of classifying textual data according to the user's intentions. Every user engagement, has a goal or an intention. Whether they desire to buy something, get more information, etc. Essentially, an intent classifier examines words automatically and groups them into intents, such as Order, Decline, Withdraw, etc. This helps to obtain helpful information, automate procedures, and comprehend the motivations behind the customer inquiries. Intent classification models are learned through text examples and their corresponding intents. A representative sample of training data is a set of raw data that has been manually labeled. After the model has been learned, the words or expressions of the user's input can automatically be linked to a specific intent through intent classification. For instance, a machine learning model can discover that terms like 'Hello' and 'Hi' are frequently associated to the intent of 'Greetings'.
Intent classification is a crucial part of chatbot's architecture.

Entity extraction [5], also known as entity recognition is a method for extracting information from text through recognizing its essential components and grouping them into predetermined categories. Based on definitions of words and context, this is the process of separating sentences into clauses, such as nouns, verbs, adjectives, and adverbs. As an illustration, in the sentence "I need a doctor, to bring me back to life", the intent is "need", while the entity is "a doctor, to bring me back to life". An entity is the relevant information of the interaction

2.2.3 Dialog Management

The dialog management component manages and updates the context of conversation. All of the information that a system needs to connect the present interaction to the overall conversational picture is referred to as context data. During the conversation it maintains the present intent and the identified entities. If the chatbot is unable to gather the required context data, it approaches the user for extra context data to fill in any gaps. As an example, if someone is buying chocolate and the seller asks ”do you want black chocolate” and you say ”yes,” the seller is aware of your interpretation of ”yes” in the context of the question.

The dialog management component is typically made up of three modules:

- *The ambiguity handling module.* When the chatbot is unable to determine an intent or when no input is detected, this module provides replies. Even if the user asks a most uncertain question, the chatbot can satisfy him by replying that it did not have an answer, asking for clarification, starting a new discussion, or providing a generic response that addresses a range of topics.
- *The data handling module.*Stores user data in a file. This way the chatbot can adapt its responses to a specific user, or become more intelligent, through time.
- *The error handling module.* Users will not always proceed according to the predetermined conversation flow. They can attempt to interrupt the procedure to ask a question or they might just wish to end it immediately. To ensure effective chatbot operation, the error handling module deals with unforeseen issues. One option to ensure the conversation flow is by implementing interruption handler which will be checking for interruptions on every turn, on every new received message from the user.

The chatbot then moves on to the further activities, which could involve responding to the user or retrieving data from the backend.

2.2.4 Backend

Once the intents and the entities are identified, a proper response must be chosen. The chatbot gets the information that it needs from the backend, usually through API calls or database queries. After the information is obtained, it is sent to the Response Generation Module and finally to the Dialog Management Module. Rule-based chatbots typically use a list of handwritten responses that relate to the user’s inputs. An AI-based chatbot would learn such mappings from existing dialogs.

For a chatbot to be effective, the knowledge base must address a wide range of user inquiries and provide a variety of answers for the same user input, which prevents duplication of replies. A chatbot may employ a relational database [12] to remember previous exchanges and so improve the consistency and relevance of the message. This approach makes the dialog more precise and consistent.

2.2.5 Response Generation

The response generation component generates responses using one or more of the three types of models rule-based, retrieval-based, and generative-based. The rule-based model chooses the response from a list of rules instead of creating brand-new text responses. The response generation module receives the placeholder data from the dialog management component and fills up the response template as necessary. The knowledge base used by rule-based models is structured using conversational patterns.

The retrieval-based model is more adaptable since it chooses the best response after checking and analyzing the resources that are available through APIs. These models know how to provide the best response from a predefined responses in a database. Based on the most recent and prior inputs, a generative model employs NLG to respond in a natural language that is human-like. It is also possible to use hybrid approaches that contrast the retrieved and created responses and pick the best one. After creating a response, the chatbot displays it to the user and waits for the next input.

3 Commercial chatbot tools

A variety of tools for building chatbots is available. Among these many do not require programming, so a user can create his own chatbot, including conversational AI-based chatbots without the knowledge of programming. In this chapter we review some of the most popular tools.

3.1 Microsoft's Bot Framework

This Bot Framework includes a modular and extensible SDK for building bots, as well as tools, templates, and related AI services. Developers can use this framework to create bots that use voice, understand natural language, manage questions and answers. Microsoft Bot Framework supports interaction through SMS, Skype, Slack, Email, Office 365, Twitter, and Telegram. The framework is made up of two major components: their Bot Builder SDK and their 'LUIS' NLU system. Its Bot Builder SDK is compatible with .NET and Node.js programming frameworks. LUIS claims to support over 30 languages with its automatic translation feature. It includes a variety of samples and templates to enable developers create bots. Nevertheless, the Microsoft bot framework has some limitations. For example, doesn't support a wide set of programming languages, but only supports Node.js and C#. It is therefore not possible to use it to develop a chatbot in another programming language if we want ot use all of the features supported by this platform. The use of other languages such as Java and Python is possible but many of the features will not be supported. Another significant drawback is the lack of support for Linux and OS X. Furthermore, the Microsoft Bot Framework does not support natural language processing, so it is not possible to create a chatbot that understands human speech. This means that the developer must process user input using a separate NLP toolkit before it can be used by the Microsoft bot framework. Also, some popular messaging platforms, such as WhatsApp and Facebook Messenger, are not supported.

3.2 Dialogflow

Dialogflow is a builder of natural language, conversation-based human-computer interaction technology, which is owned by Google. This platform is a viable choice for creating chatbots that can be integrated into websites or mobile applications. Produce customer experiences with virtual agents that facilitate conversations, built with Google Assistant's technologies, and speeding up the development process by using interactive flow visualizations that enable builders to quickly see, recognize, modify, and share their experience.

This platform also has certaion limitations. It is highly unlikely that the same intent exists in different contexts. It is difficult to control the flow of the conversation, and the bot frequently misinterprets the user's requests. Synonym and hyponym understanding is limited. The platform can only process the sentence, but not understand it. There would also be a problem if someone want to support a language that DialogFlow does not support.

3.3 Botsify

Botsify is a chatbot builder that is also intended for Facebook users. Using this machine learning-based programming platform, one can create AI chatbots for a custom websites or for a Facebook Messenger. Their user interface works on a simple drag-and-drop principle which allows users to create their own templates. A number of templates, such as hospitals, travel agencies, dining establishments, and basic FAQs are already provided by the platform so users can choose from these, or build their own. They employ highly personalized and perceptive conversational forms that make it easy to save user data in spreadsheets or in a CRM. The techniques, tactics, and technology used by businesses to analyze and manage data and customer interactions collectively are known as customer relationship management or CRM. CRM systems gather information about customers from various points of contact between them and the business. An artificial intelligence-powered Botsify bot can also be created. This requires a large database of questions and answers to train the bot. A chatbot can be trained incrementally by providing additional data. With each session, the bot gains more knowledge and becomes more intelligent in its responses. Customizing the questions and flows allows users of this platform to select which queries should be addressed by the chatbot and which by a human. It offers a possibility of switching to a human agent, but only with the help of email or push notifications on a browser or a mobile device. An artificially intelligent chatbot from Botsify searches its knowledge base for the best answer after analyzing user questions for similar terms and phrases.

After we tested some of their demo versions, it turned out that the chatbot builder displays a lot of error messages, which are obviously caused by the chatbot's limitations rather than user errors. The technology is static and has significantly lagged behind chatbots that use AI. It doesn't seem as an AI-powered chatbot solution, for it can only use automatic keyword replies. if he wanted to give the chatbot some intelligence we would have to use Dialogflow's interaction with Botsify. This would not be a practical AI chatbot choice if there is no included dedicated developer to program and manage the Dialogflow bot because it requires a lot of human labor, time and training.

3.4 Botpress

Botpress is an open-source conversational AI program, tool for building chatbots in a drag-and-drop manner through the use of visual flows. It offers a built-in NLP engine with spell checking, entity extraction and intent classification capabilities. It includes a debugger and simulator of conversations. Botpress offers possibility for integration with the most well-known chat platforms like Telegram, Microsoft Teams, Slack, and Messenger. Users can choose their open-source free plan or the enterprise plan which is paid. With the paid plan, Botpress supports 12 different different languages that are not available with the free version.

3.5 IBM Watson Assistant

IBM Watson Assistant is a cloud service that enables enterprise-level software developers to include an artificial intelligence virtual assistant into the product they are creating and give the assistant their own brand. It can be used with the IBM Cloud to deliver the service, which allows access to Watson AI. The Assistant employs state-of-the-art AI to comprehend inquiries that the clients ask in plain English. To provide precise responses in real time, it leverages machine learning models that are specially created from the data. Through a process known as ”autolearning,” Watson is built to pick up knowledge from every discussion and improve itself automatically over time. Watson responds to client questions with the most pertinent information possible, enhancing its abilities without human oversight. Watson Assistant enables non-technical people to create, train, and manage sophisticated chatbots, without writing program code. Watson Assistant gives users access to a powerful analytics dashboard that enables them to identify patterns, comprehend areas of friction, and act right away to enhance how their helpers function. Watson may be deployed on any public cloud, including those from IBM, Amazon, Google, and Microsoft, as well as on-premises with IBM Cloud Pak for Data or a private cloud.

According to our analysis of this service, there are some restrictions and weaknesses with this service. Because of poor documentation and the chatbot’s inability to comprehend some questions, real-time conversations are often unreliable. Many users requested that additional coding compatibility should be added for developers because it can be quite challenging to find a quick fix for any fault or issue.

3.6 Conclusions

There are a lot of options on the internet, from open source to commercial platforms, where buying a license for creating chatbot is required. However, there are some good reasons we may want to build our own chatbot from scratch. We may not want to send private information to a third party. Many people don't trust cloud companies, or it takes a lot of work to cover compliance when delivering material that potentially contain personally identifiable information. Available products may not support the language we want to use. Additionally, the language may have a very particular vocabulary that is difficult for a standard natural language parser to handle. For these reasons we will consider how to build a custom chatbot from scratch.

We also have an additional motivation, which is to learn about chatbot architecture from a particular perspective. Our chatbot will be able to answer student questions and give support relation student affairs issues.. In cases when our chatbot is not able to find the right answer to the user's question, we will add an option to connect the user to an operator, so the conversation flow will not be interrupted. Our work will include building a web user interface, a client-server communication protocol and the complete server-side chatbot architecture.

4 Implementation

In this chapter we will present our implementation of a chatbot and the tools that we used. This chatbot is targeted towards providing help to students about questions regarding students affairs. The chatbot is integrated into a web page. The web page and the chatbot are both developed in the JavaScript programming language, using open source framework for web development known as Next.js, which is built on top of Node.js, and uses React.js, a free open source library for building user interfaces, for the frontend. To find answers to students questions we use two ML models that are implemented in the JavaScript version of Tensorflow. This Tensorflow library allows to use ML models directly in the browser or in NodeJS. The questions that the chatbot cannot answer, are stored in a database, so the admin can later review and use them to improve the ML model. We use MongoDB database managament system. Communication between the client and the server is implemented using the Socket.IO library, which is a part of the NodeJS framework. In the case where questions can not be answered by the chatbot, user can decide if he wants to connect to a human. The operator will then have full control of the conversation and provide the user with information he needs.

4.1 Requirements Analysis

This section presents the analysis our system requirements. The requirements are structured in a functional and non-functional part. In the functional part we specify what our system must be able to perform. These are the functions that help users complete their tasks. The non-functional requirements are about the speed of the system, the security of the data, compatibility, reliability and how to ensure the effectiveness of our system.

4.1.1 Functional requirements

This system needs to offer the basic functions to our users. The system is presented as a web page with a chatbot, so every user should be able to get in touch with the chatbot through this web page and communicate with the AI behind the chatbot. The user should be able to send messages to the chatbot. Furthermore, if the user cannot get the response from our chatbot AI he must be able to establish a connection with an operator. The operator is a human, with the role to communicate with the web page visitors through a simple chat window. The system should allow the operator to access the wider set of functionalities of the web page associated with running the conversations with the web page visitors.

In figure 1 we present a use case diagram of our system with all functional requirements for all actors that takes part in our system.

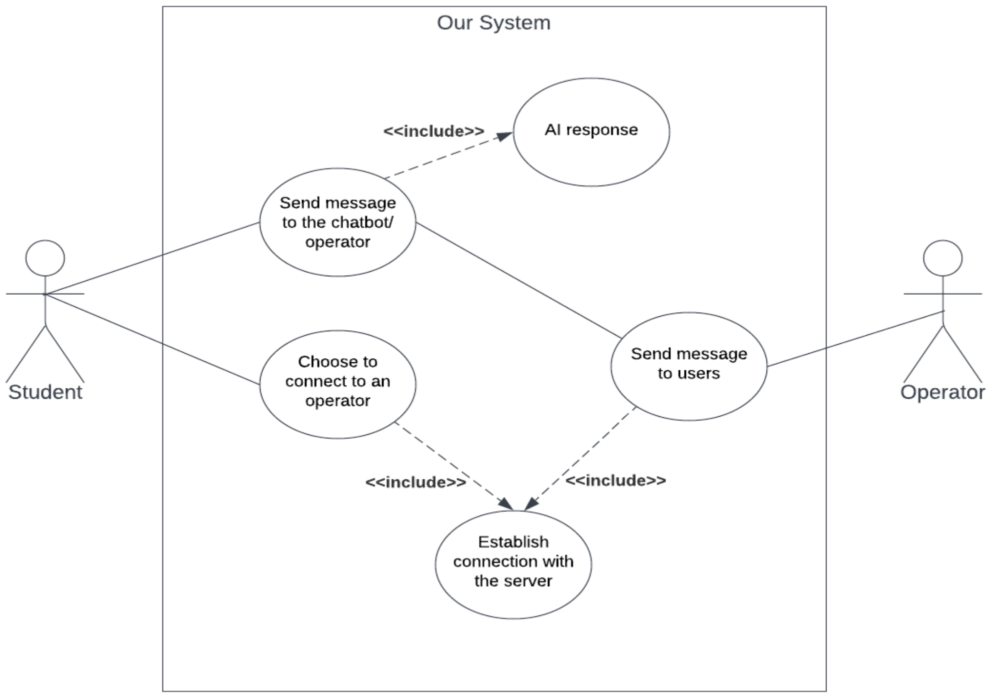

Our System
<<include>>
AI response
Send message to the chatbot/ operator
Student
Choose to connect to an operator
Send message to users
Operator
<<include>>
<<include>>
Establish connection with the server

Figure 1: Use Case Diagram

A student actor can send messages to the chatbot or choose to connect to an operator when the chatbot AI is not able to give an answer. The operator is able to manage the conversations through his chat window when the users are routed to him.

4.1.2 Non-Functional requirements

For the chatbot to understand the inquiries, it must be able to understand the English language. The chatbot's response time should be shorter than the time it takes the human to write a response. The chatbot was designed to inform students about the university matters, therefore the conversation should always be in that direction. The chatbot must make its aim clear in the conversation and avoid misleading the user. Chatbot must be able to comprehend natural language within a specific domain and be able to identify the topic of the user's input, regardless to the sentence structure. Following the user's input, this chatbot should be able to choose or generate a response, depending on that information and the conversation's context. The provided responses must make sense in the context inferred from the entry. The system should be made scalable, so that it can accommodate many users simultaneously.

4.2 System architecture

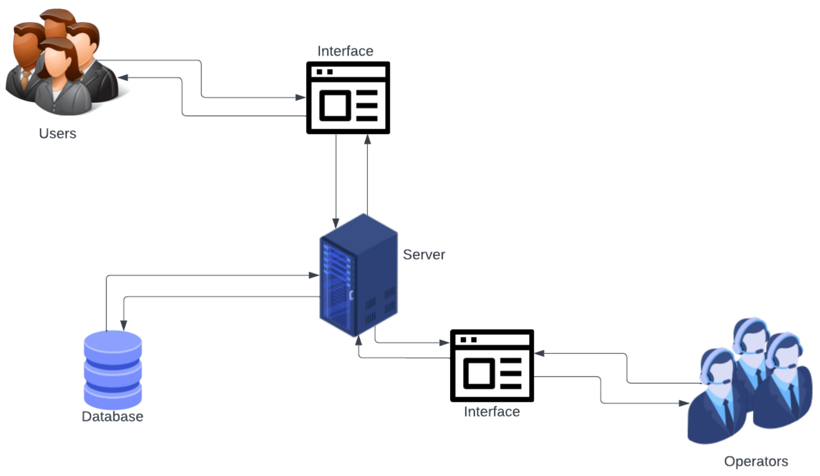


Figure 2: Schema of our system architecture

A simple schema of our system architecture is shown in Figure 2. Users get in touch with our system through their user interface. They first visit our web page and than communicate with the chatbot. If the chatbot cannot find an answer to the given question, then, on the user's request, a connection with the operator is established. Operators on the other side are able to see through their interface all the users that are connected to the server. Messages can be send both ways, while the server takes care of routing the messages to the correct receiver. The database is used to store the questions that cannot be answered by the chatbot. These messages are later used to further improve the chatbot.

4.2.1 Client-server model

Our architecture is based on the client-server model. The client-server model is a type of architecture that has a client, a device that can make requests over the internet, and a server that serves the responses to the incoming request from the client. A client-side application, used by a user who wants to get in touch with our system, must first send a HTTP request to the server. The client-side application waits for the HTTP response, which is in the form of a HTML document. While rendering the web page on the client-side application, the ML models are loaded. Chatbot is now available so the user is able to start an interaction. If the user wants to ask a question, he types it in and with the help of the ML models, the question is getting processed. If the ML models can find an answer to the question, the answer will be displayed. If an answer is not found, another message that allows the user to connect to an operator is displayed. The client-side application then establishes a bidirectional connection with the server, using the WebSocket protocol. The operator, who is already connected to the server, receives the user's request for help. The questions that have not been resolved by the chatbot are automatically stored to the database.

When a HTTP request is sent by a client, the server is responsible for preparing the response and delivering it. The server distinguishes between two types of clients - students and the operators. Every communication between a student and an operator is done over the server, which routes the sent messages to their intended receivers. The server also communicates with the database to store the messages.

4.2.2 Deployment diagram

Deployment diagram describes the physical deployment of the hardware components. In figure 3, our system's deployment diagram is shown. There are three main parts: the client, the server, and the database. The client sends a request to the server, which processes the request and sends back the response. The user on the client side can type a question, the ML models loaded on the server side process the question and give the answer to the user. If that question is not understood, the server invokes an operation for inserting that question into the database.

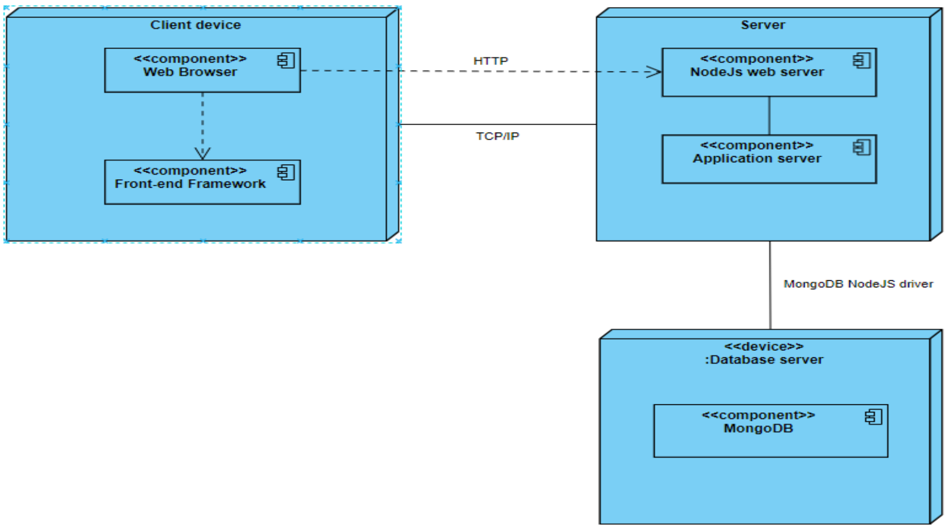

Client device
<<component>> Web Browser
<<component>> Front-end Framework
HTTP
TCP/IP
Server
<<component>> NodeJs web server
<<component>> Application server
MongoDB NodeJS driver
<<device>> :Database server
<<component>> MongoDB

Figure 3: Deployment Diagram

Database
We are using an open source database called MongoDB which is type of database management system that stores the data in JSON format. This database is adaptable, allowing differences in document structure and can also store incomplete documents. The connection between the NodeJS server and the MongoDB is established through the MongoDB NodeJS driver.

4.2.3 Technologies used

This section highlights the technologies that we used in our implementation.

NodeJS is a run-time environment that enables JavaScript programs to run outside a web browser. NodeJS is mostly used for server-side scripting.

ReactJS is a client-side JavaScript library. It works with reusable UI components that lets the developer divide the app into discrete sections which operate separately from each other.

NextJS is a framework that supports server-side development based on NodeJS and client-side development based on ReactJS. It has built-in data fetching mechanisms and server-side rendering and as well as intelligent page navigation. Our implementation is done in this framework.

TensorFlowJS is a JavaScript machine learning that enables deploying ML models in the web browser. With this library we can integrate machine learning features into any web application.

SocketIO is a library that facilitates event-based, bidirectional communication between a server and a client. It is based on the WebSocket protocol, and for the communication to be possible, both the client and the server must use the Socket.IO package. The data can be transmitted in a variety of formats. This library is used for communication between the operator and the user of the chatbot.

4.3 Chatbot implementation

The inteligence behind our chatbot is based on two ML models that helps each other in determining a meaningful response to the user's request. BOne is called the *Universal Sentence Encoder* model and the other one the *Bert model*. Both are providing their answers. To provide a well defined answer, both must be used. Reasons for that are explained at the end of this section.

4.3.1 BERT and the transformer

For typical natural language processing tasks, one specific neural network architecture has demonstrated to be particularly effective - the model called *transformer*. Transformer is a model that uses an encoder and a decoder to change one sequence into another based on the help of the attention mechanism, also called an *attention model*. A neural network called *sequence-to-sequence* may change one sequence of elements, such as the words in a phrase, into another sequence. The input sequence is transferred into a higher dimensional space by an encoder. The abstract vector is then sent to an encoder, which converts it into an output sequence. A mechanism called the *attention* analyzes the input sequence and determines which additional sequence elements are crucial at each stage. The attention mechanism simultaneously considers all the inputs that the encoder reads, and determines which inputs are significant by assigning various weights to those inputs. The encoded sentence and the weights produced by this attention mechanism are taken by the decoder to improve the decoding process and its output.

The Bidirectional Encoder Representations from Transformers or BERT is a NLP model, where bidirectional means that it reads the input both ways, from right to left and left to right. This enables it to better understand the context of a word by considering all its surroundings. A BERT model uses two different training strategies, one called *Masked Language Modeling* and the other *Next Sentence Prediction*.

Masked Language Modeling

In Masked Language Modeling (MLM), a specific portion of the words in a sentence are often hidden, and the model based on the other words is supposed to predict those hidden words. Because of the bidirectional approach, the masked word is learned based on the words that appear on it's left as well as the right side. A simple example of predicting the masked word in a sentence based on the other given words through the MLM scheme is shown in Figure 4.4. In the sentence ”All the very best” the word ”very” is masked out and the masked sentence is given to the MLM. The MLM predicts the masked word to be ”very”.

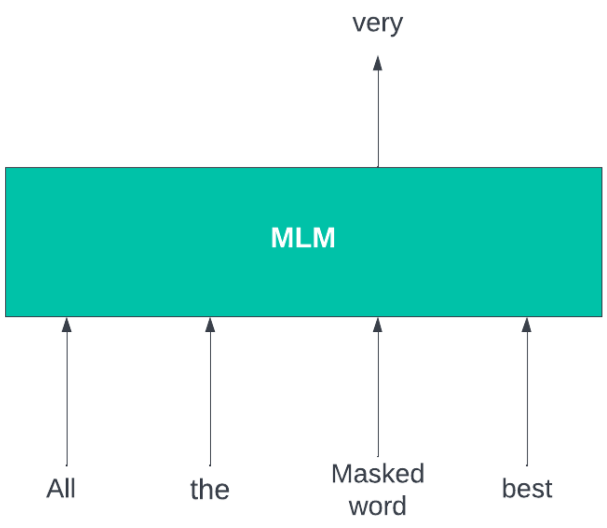


Figure 4: Masked Language Model

Next Sentence Prediction

Language modeling does not adequately capture the link between two sentences, which is necessary for many crucial tasks like answering a question. A separate model is used to predict whether the second sentence in a pair will come after another. The model receives two sentences and evaluates the probability of the second sentence to appear after the first one in the original text.

4.3.2 The training dataset

The BERT model that our system uses is a model which is is able to respond to queries based on the content of a passage. The passage is a text from which the model finds the answer to the question, of course if the question is answerable from that passage. We used the Stanford Question Answering Dataset, a data set consisting of questions based on Wikipedia articles to train our model. We used this data set not only to test the ability of our model to answer questions, but also to abstain from questions that can not be answered. Our model was trained with the data set that we created ourselves and consists of passages that contain the information about student affairs at Famnit.

4.3.3 Universal sentence encoder model

For tasks involving sentence classification and sentence similarity, the Universal Sentence Encoder is employed. Universal sentence encoder model transforms a freely written text into a numerical vector. This process is called text embedding. After a sentence is converted to a vector the similarity is calculated based on the distance between these vectors.

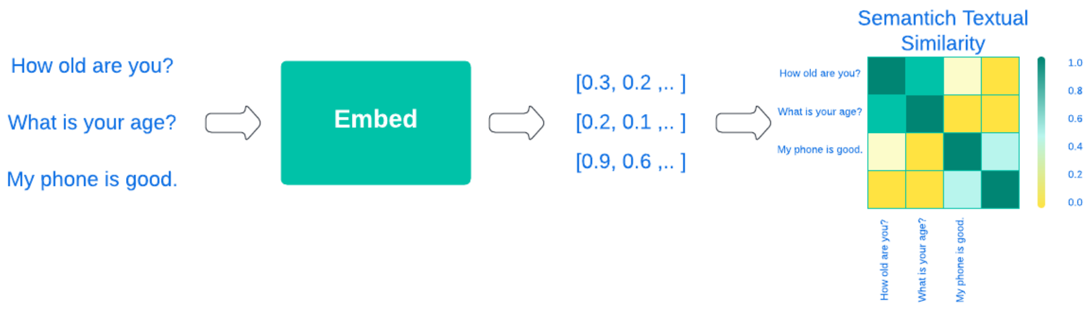


Figure 5: Semantic Similarity

Figure 5 shows an example of how semantic similarity is evaluated through text embedding. First, the sentences are embedded, converted into vector and then calculated their distance between each other presented in a matrix.

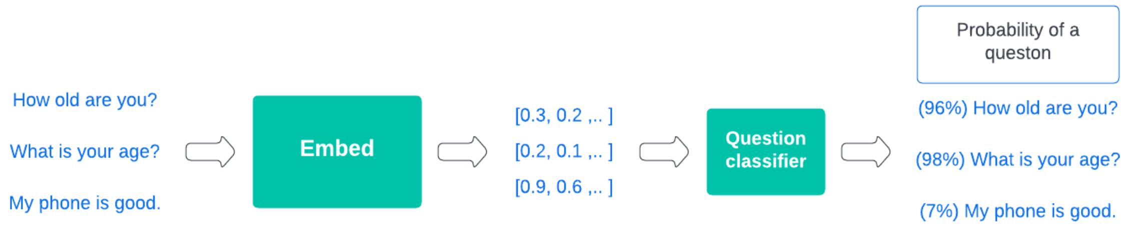


Figure 6: Classification

Labeling or grouping text data into categories is a technique called text classification. Figure 6 depicts the process of text classification. It shows the classification of a sentence into classes that represent the probability of this sentence being a question. After the sentences are embedded, they are classified using the question classifier.

4.3.4 Final improvements

While testing our initial models, we realized that we need a better representation of answers. In our case, the BERT model, which works on a given passage, is checking all the sentences in that passage and the answer that is found can be a word or a part of a sentence, but not the whole sentence. We wanted to give to the user a well defined answer. For this reason we split the text into sentences and then these sentences are input to the Universal Sentence Encoder model. When the BERT model finds an answer from the passage, we show the answer from the Universal Sentence Encoder model, which is represented as whole sentence. In the case when our BERT model can not find an answer, we give to the user the option to establish a connection with the operator.

The reason to process the same question with both models is explained in Figures 7 and 8.

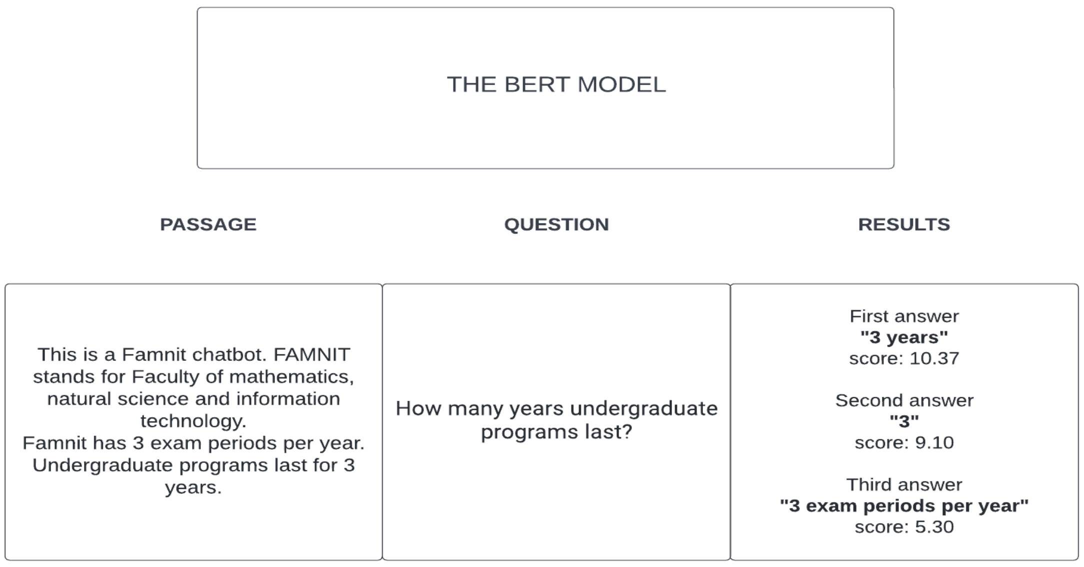

THE BERT MODEL
PASSAGE
QUESTION
RESULTS
This is a Famnit chatbot. FAMNIT stands for Faculty of mathematics, natural science and information technology. Famnit has 3 exam periods per year. Undergraduate programs last for 3 years.
How many years undergraduate programs last?
First answer "3 years" score: 10.37
Second answer "3" score: 9.10
Third answer "3 exam periods per year" score: 5.30

Figure 7: Bert model example

Figure 7 shows three answers given by the BERT model. These answers do not satisfy our notion of a well defined answer. The answers are containing just one or a few words, while we want to reply to the user with a whole sentence. If this model can not find an answer in the passage for the given question, it will not return any result. The response will be empty.

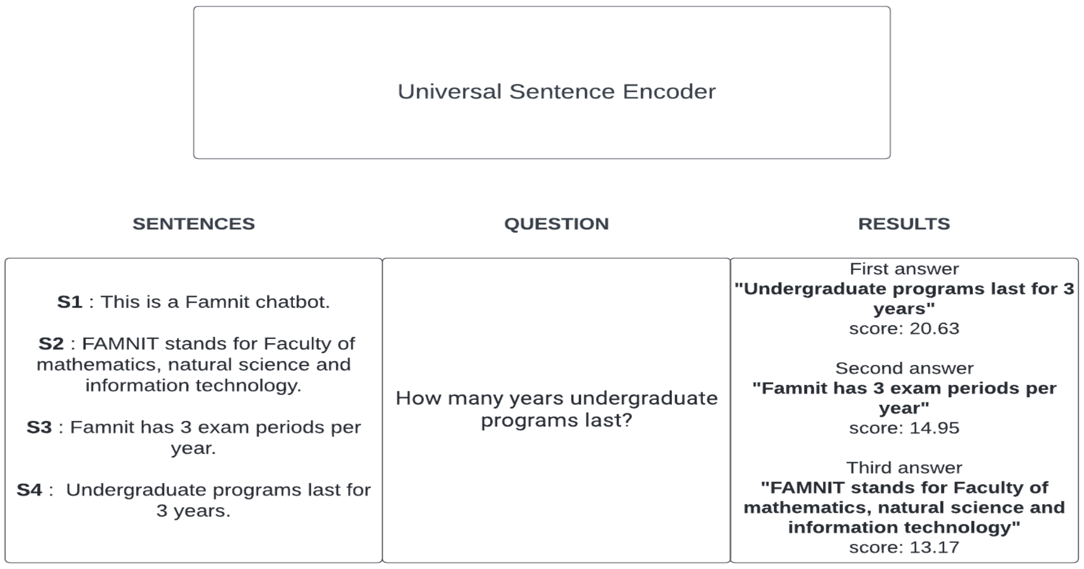

Universal Sentence Encoder
SENTENCES
QUESTION
RESULTS
S1 : This is a Famnit chatbot.
S2 : FAMNIT stands for Faculty of mathematics, natural science and information technology.
S3 : Famnit has 3 exam periods per year.
S4 : Undergraduate programs last for 3 years.
How many years undergraduate programs last?
First answer "Undergraduate programs last for 3 years" score: 20.63
Second answer "Famnit has 3 exam periods per year" score: 14.95
Third answer "FAMNIT stands for Faculty of mathematics, natural science and information technology" score: 13.17

Figure 8: Universal Sentence Encoder example

Figure 8 shows the output for the same sentence by the Universal Sentence Encoder. Here, as a result, we get a list of whole sentences starting from the most rated one, which in this case is the right answer to the question. We can see that the second and the third option are not even close to the right answer, but they are still formulated as complete sentences. We now simply select the sentence with a high score, which is the most similar to a highly scored short answer from the BERT model.

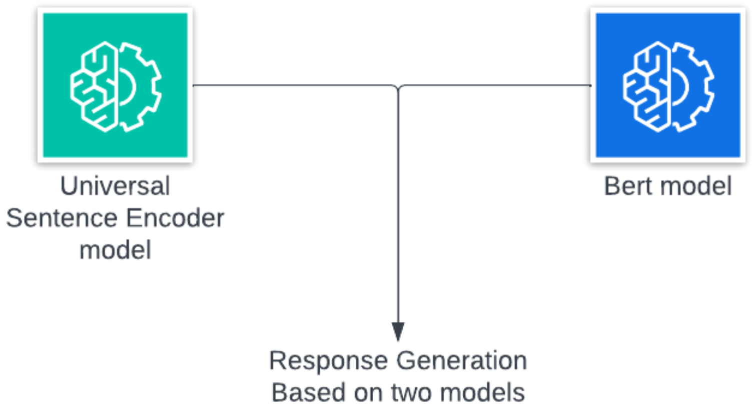

Universal Sentence Encoder model
Bert model
Response Generation Based on two models

Figure 9: Conversation flow

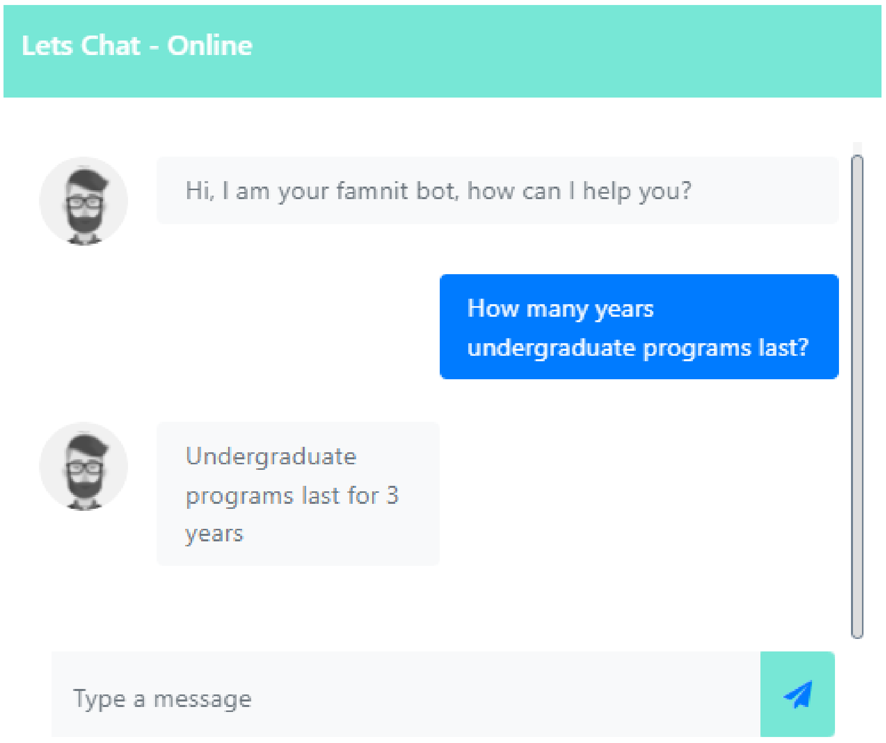
The interaction between the Universal Sentence Encoder model and the Bert model is shown in Figure 9.

4.4 Results

In this section we present practical scenarios of our implementation and demonstrate how our chatbot works.

4.4.1 Sample scenarios

The following figures are illustrating a conversation between the chatbot and a user. The user is asking questions and the chatbot responds.

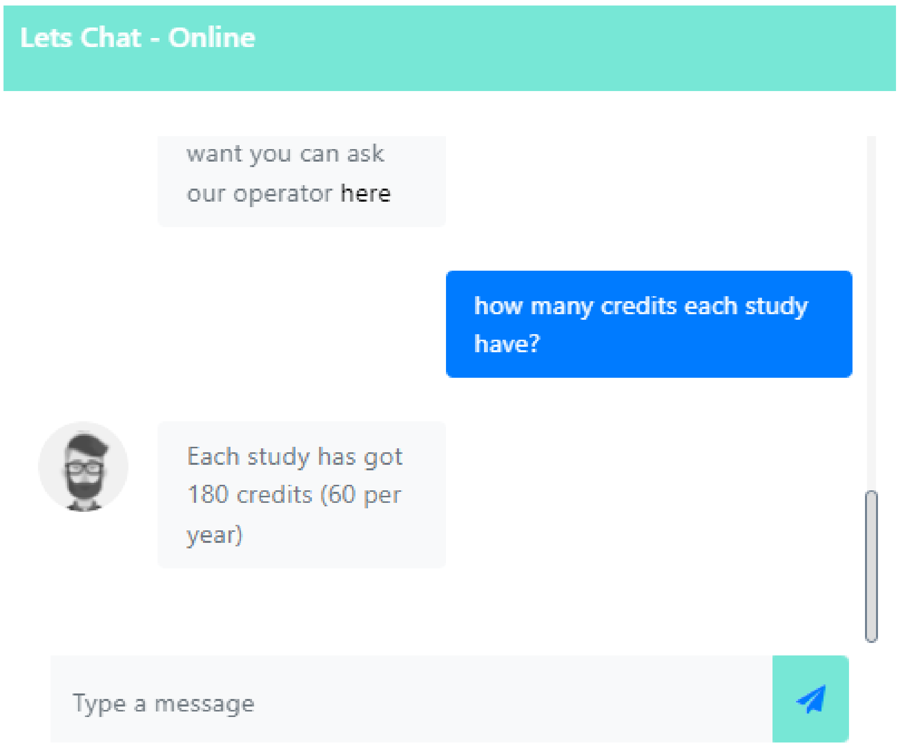

Lets Chat - Online

Hi, I am your famnit bot, how can I help you?

How many years undergraduate programs last?

Undergraduate programs last for 3 years

Figure 10: Interaction between the chatbot and the user.

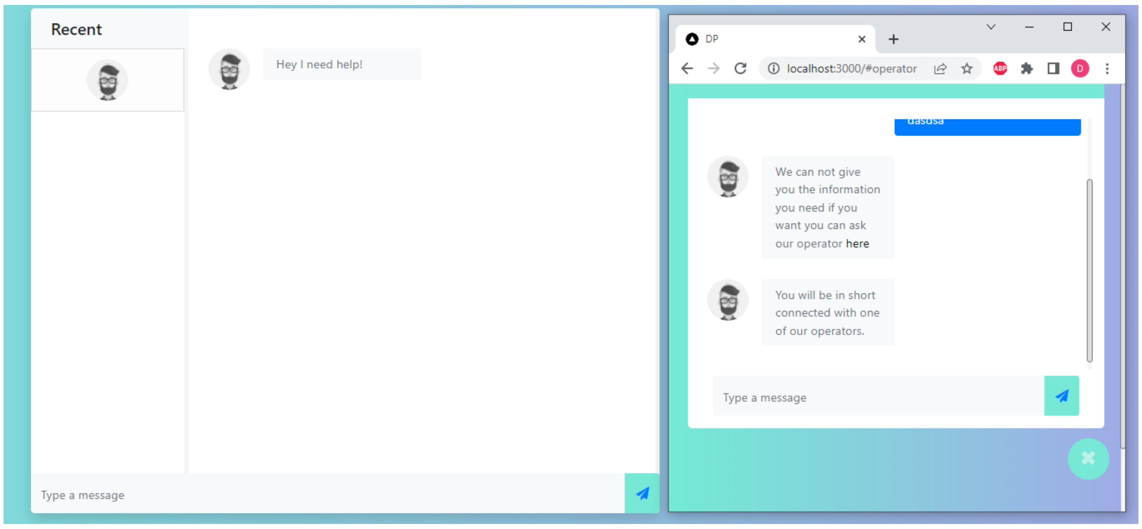

Lets Chat - Online

want you can ask our operator here

how many credits each study have?

Each study has got 180 credits (60 per year)

Figure 11: Interaction between the chatbot and the user.


Recent

Hey I need help!

Type a message

DP

localhost:3000/#operator

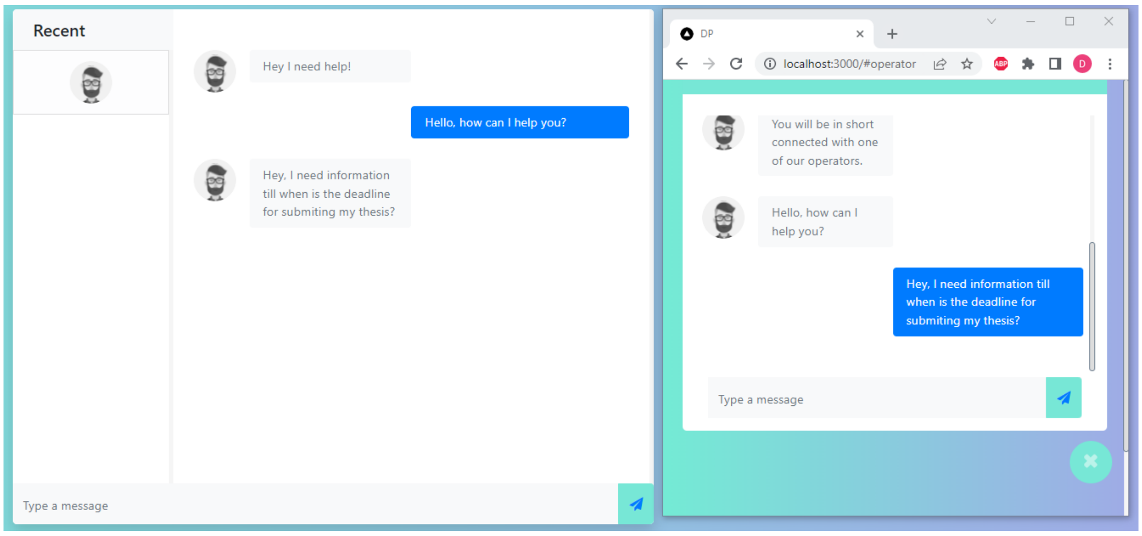
We can not give you the information you need if you want you can ask our operator here

You will be in short connected with one of our operators.

Type a message

Figure 12: Establishing the connection with an operator.

Figure 12 shows a user establishing a connection with the operator after the ML model wasn't able to answer the question.


Recent

Hey I need help!

Hello, how can I help you?

Hey, I need information till when is the deadline for submiting my thesis?

Type a message

DP

localhost:3000/#operator

You will be in short connected with one of our operators.

Hello, how can I help you?

Hey, I need information till when is the deadline for submiting my thesis?

Type a message

Figure 13: Communication between the user and the operator

Figure 13 shows a communication between the user on one side and the operator on the other, through the operator's panel.

5 Conclusion

This thesis outlined a typical architecture of a chatbot, reviewed existing solutions and implemented our own chatbot system specialized for answering students question at our faculty.

After reviewing the existing solutions, we noticed that there are a lot options, from free to commercial for creating a chatbot. Based on the review and the found limitations and disadvantages of these platforms we concluded that there are good reasons to build our own chatbot, starting from scratch. These platforms use the private information of their users and we don't want to share the information with a third party. Some of the available platforms do not support the language that we want to use and some of them are not using a good AI backend for generating the responses and maintain the conversation flow. After the review, we chose the of technologies to use and discussed why these technologies are ideal for this kind of task. We studied the literature about AI and ML and settled for a viable approach. Through the development of this solution we faced some challenges in the part of natural processing and generating the answers. Our AI chatbot was based on only one model. After the first testing, the results were not satisfying. Generated responses were not well defined, because they were not formed as whole sentences. We then incorporated the second ML model called the Universal Sentence Encoder. We defined a new function for providing a response from both models. We described the development process in the final chapter of this thesis and gave examples on how our solution works.

As a part of the possible future work, the support for multiple languages could be considered, especially due to the international orientation of our university. For now, only English language is supported. A different dynamic version of the chatbot may be created, and more difficult user issues could be examined. If the user is unable to provide sufficient clarification, the chatbot could recommend potential difficulties to the user in the form of a option list with the basic problems rather than directly requesting a human operator.

6 Povzetek naloge v slovenskem jeziku

Interakcija med računalnikom in človekom zavzema pomembno mesto v tehnološkem razvoju. Na tem področju lahko z razvojem sofisticiranih modelov za razumevanje naravnega jezika veliko pripomore umetna inteligenca oz. strojno učenje. Lep primer sinergije umetne inteligence in interakcije računalnik-človek so t.i. *klepetalniki* oz. chatbot-i.

Ta diplomska naloga opisuje sodobno strategijo dostopa do informacij, ki se uporablja za izboljšanje izobraževanja in ima obliko pisnega dialoga. V tej nalogi predstavimo implementacijo sistema za pomoč študentom s spletnim klepetalnim robotom, ki študentu pomaga pri iskanju odgovorov na vprašanja, povezana s študijem na Fakulteti za matematiko, naravoslovje in informacijske tehnologije.

V prvem poglavju definiramo problem in cilje diplomske naloge. V drugem poglavju razložimo kako klepetalniki delujejo, kako poteka obdelava naravnega jezika, kako se generirajo odgovori in opišemo arhitekturo klepetalnikov. Predstavimo tri kategorije klepetalnikov — klepetalnike, ki uporabljajo vnaprej določenen niza pravil, umetno inteligenco ali oboje. Sledi pregled obstoječih orodij za kreiranje klepetalnikov, kjer ugotavljamo njihove prednosti in slabosti. Izkaže se, da je na voljo veliko različnih plačljivih in brezplačnih možnosti, kljub temu pa imamo jasne razloge za implementacijo lastnega klepetalnika. Ugotovili smo namreč, da nekatere platforme zbirajo in uporabljajo osebne podatke svojih uporabnikov, kar je v nasprotju z našimi interesi. Nekatera od obstoječih orodij za generiranje odgovorov na vprašanja uporabnikov ne uporabljajo umetne inteligence oz. ne v meri, ki bi bila zadovoljiva za naše potrebe.

V nadaljevanju predstavimo lastno implementacijo klepetalnika. Najprej izberemo ustrezne programske tehnologije ter navedemo razloge za naš izbor. Nato povzamemo teoretične osnove delovanja klepetalnikov, ki so osnovani na umetni inteligenci. Sem spadajo tudi izzivi tolmačenja naravnega jezika, njegovega razumevanja in generiranja odgovorov. Nap klepetalnik je najprej temeljil na enem modelu strojnega učenja, kar pa ni dalo zadovoljivih rezultatov. Klepetalnikovi odgovori niso bili vedno v obliki celotnih stavek, ampak se je včasih zgodilo, da je bil odgovor podan samo z eno besedo. Ta problem smo rešili tako, da smo dodali še drugi model, ki deluje kot *univerzalni kodirnik stavkov*. To je zahtevalo implementacijo funkcije za združevanje odzivov obeh modelov.

V zadnjem poglavju najprej podamo analizo zahtev našega sistema, nato sledi celoten pregled razvojnega procesa naše rešitve, kjer formalno opišemo, kako naša rešitev deluje. Predstavimo tudi oba modela, njuno delovanje in primere procesiranja sporočil ter odgovorov na vprašanja. Predstavimo izgled uporabniškega vmesnika za osebni računalnik in mobilne platforme. Uporabniški vmesnik prikazuje scenarije postavljanja vprašanj in prejema odgovora ter vzpostavitve povezave z operatorjem, v primeru da klepetalnik ni znal odgovoriti na uporabnikovo vprašanje.

Diplomsko nalogo zaključimo z nekaj idejami za prihodnje izboljšave. Ena od idej je podpora za več jezikov, saj klepetalnik sedaj razume le angleški jezik. Klepetalnik bi lahko nadgradili tudi tako, da v primeru, ko ne razume uporabnikovega vprašanja, uporabniku ponudi seznam možnosti, preko katerih uporabnik vprašanje sestavi. Tako bi lahko nekoliko razbremenili delo človeškega operaterja. Takšne spremembe smo predvideli že med izdelavo klepetalnika in ga zasnovali na fleksibilen način, ki omogoča enostavno dodajanje nadgradenj.

7 Bibliography

[1] Nikita Bachani. Chunking in nlp: decoded. `https://towardsdatascience.com/chunking-in-nlp-decoded-b4a71b2b4e24`, 2022.

[2] Vikas Bhandary. Development of social chatbots. `https://towardsdatascience.com/development-of-social-chatbots-a411d11e5def`, 2020.

[3] commbox. How can chatbots improve customer service? `https://www.commbox.io/how-can-chatbots-improve-customer-service/`.

[4] Sarada Devaram. Empathic chatbot: Emotional intelligence for empathic chatbot: Emotional intelligence for mental health well-being, 2020.

[5] Anna Diaz. Nlp – what happens in entity extraction and its value. `https://www.marscrowd.com/blog/text/nlp-entity-extraction-and-its-value/`, 2021.

[6] Andrew Freed. *Conversational AI*. Simon and Schuster, 2021.

[7] Aurelien Geron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems 2nd Edition*. O'Reilly Media, 2017.

[8] Imperson. 3 ways chatbots are transforming the entertainment industry. `https://www.imperson.com/post/3-ways-chatbots-are-transforming-the-entertainment-industry`, 2021.

[9] Christopher Marshall. What is intent recognition and how can i use it? `https://medium.com/mysuperai/what-is-intent-recognition-and-how-can-i-use-it-9ceb35055c4f`, 2020.

[10] Chinedu Wilfred Okonkwo and Abejide Ade-Ibijola. Chatbots applications in education: A systematic review. *Computers and Education: Artificial Intelligence*, 2:100033, 2021.

[11] Gil Press. Ai stats news: 86% of consumers prefer humans to chatbots. `https://www.forbes.com/sites/gilpress/2019/10/02/ai-stats-news-86-of-consumers-prefer-to-interact,-with-a-human-agent-rather-than-a-chatbot/?sh=25fe16b12d3b`, 2019.

[12] Prashant Chettri Rebika Rai. Relational database. `https://www.sciencedirect.com/topics/computer-science/relational-database`, 2018.

[13] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach, 4th edition*. Pearson, 2021.

[14] Peiru Teo. Ai chatbots in healthcare. `https://keyreply.com/blog/healthcare-chatbots/`.

[15] Charles Waghmare. *Business Benefits of Using Chatbots*, pages 147–165. Apress, Berkeley, CA, 2019.

[16] Orinna Weaver. Marketing chatbots: The marketer that's always on. `https://www.intercom.com/blog/chatbot-marketing/`.