A Natural Language Processing Framework for Training a Neural Network Chatbot

Kamila Michel

Maciej Majchrzak

B.Sc.(Hons) in Software Development

Contents

List of Figures

About this project

Abstract   The goal of our main project was to adapt the knowledge that we have gained during our four years of studying at GMIT as well as expanding it with issues that were unknown to us. We put a lot of emphasis on research on artificial intelligence. It is increasingly common opinion that artificial intelligence will completely change the world. We are at the threshold of a revolution that will change our civilization. This technology brings enormous possibilities. The dynamic development of artificial intelligence means that it will soon become a phenomenon commonly used in marketing activities. According to TechRepublic 61 percent of businesses said they implemented AI in 2017, up from just 38 percent in 2016 [1]. Our society becomes dependent on immediate access to information. Often this time determines the decision we make, we do everything faster, more efficiently, simpler. Similarly, with customer service, we no longer wait for the store to open, to get to know his offer, we do not have to go to the restaurant to see the menu. All this information can be found on websites or fan pages. Chatbot will allow us to additionally ask questions, purchase or make an immediate reservation at any time of the day or night, without the intervention of an employee. Our goal was to familiarise ourselves with one of those smart assistance and build our own chatbot using neural networks and Natural Language Processing. The research, methodologies, used technologies, and system design and evaluation of our chatbot are described in detail in the penultimate chapter of this dissertation.

Authors   This project was developed by Kamila Michel and Maciej Majchrzak, final year students of Software Development at Galway Mayo Institute of Technology.

Acknowledgements

We would like to take this opportunity to acknowledge and thank everyone who helped us during the creation of this project.

In our opinion, the person who deserves the most thanks for help at the key moment of this project is our supervisor Mr. Gerard Harrison. Thank you for all your help and support during the very difficult moments of the development phase.

We would also like to thank to Mr. John Healy, Head Lecturer of the Applied Project and Minor Dissertation module.

We would also like to thank the lecturers and employees of Software Development for their dedication, support and unforgettable four years during which We met many wonderful people and learned new skills. It was not always easy but sacrifice and hard work paid off and it was worth it. We have experienced many interesting moments here, and most importantly, gained knowledge that will allow us to find a job We always dreamed about.

Chapter 1

Introduction

The 21st century has brought us great development of technology, practically every kind. Progress in recent years cannot be overlooked in information technologies and, above all, in the evolution of the Internet. Information to which we have access, and the form of their presentation exceed our imaginations from a few years ago. The reason for this is exponential growth in the quantity, quality and information presentation technology. During four years of our studies in software development, we had opportunities to learn new technologies and skills needed in our further career. However, there is a topic that still has many unresolved questions and is still growing; that is Artificial Intelligence. Our goal was to obtain information in this field and implement it with our project. We were looking for an idea where we could implement the knowledge we have already acquired and combine them with new problems that were not discussed in our course. The growing importance of computers in the daily life of people and their rapid development has meant that for years, attempts have been made to give them human characteristics. Implementation of something that is an extremely popular topic in science fiction literature, from year to year, it seems to be more likely. Regardless of whether these attempts are undertaken for educational purposes, or to simplify and increase cooperation between the machine and human. In all those cases an important element is the dialogue and the simplest form is the implementation of given commands. Humans personify objects, animals and phenomena. When working with an object for a long time, it begins to give more and more human traits. It is not unusual, therefore, to talk to the device, and it is often encountered. Our first idea was to build a smart speaker using the Google AIY Voice Kit [1]. After thorough analysis and implementation, we concluded that using the Google Project will be associated with a small amount of programming. Sample code was provided, and we could only modify it. That phase of our project we treated

as an experimental process. Thanks to that we found out how the Google Cloud Speech-to-Text service works, it allows converting spoken commands into text. Based on that experience, we decided to build our own chatbot which will use a neural network needed to train the conversation data. In chapter IV of our dissertation, we describe how neurons work and how Natural Language Processing was used in our project. The main purpose of our chatbot is to have a conversation with the user who can in turn choose the conversation topic from a given list. Currently, several basic types of bots are distinguished due to their use: Assistance Chatbots, Service/Action Chatbots, NLP (Natural language processing) Chatbots.

Assistance Chatbot - conducts unidirectional communication with users ("subscribers") by sending them notifications in accordance with a fixed schedule or as a result of certain events (e.g. notification of shipment). Examples of scenarios implemented by the assistance chatbot: notification by sending a package in the online store, daily weather forecast, Daily news from the country and the world, bulletin informing about new promotions, notification about the price reduction of the observed product.

Service/Action Chatbot - allows the user to go through a predetermined and linear process, requiring a series of decisions from a closed pool of choices. Currently, such chatbots do not have the NLP (Natural language processing) engine most often. Also do not allow for a free conversation with a user who is forced to choose from a narrow option menu offered to him at a given moment. Other examples of scenarios realized in this way are: the purchase of tickets to the cinema, the choice of a summer offer of a travel agency, submission of an application for opening a bank account.

NLP (Natural language processing) Chatbot - allows the user to talk freely by following instructions and answering questions asked by the user's "natural" language. It is possible thanks to the use of so-called NLP (Natural Language Processing) engine, which of the user's statements is able to pick out his intention (intent), as well as the relevant parameters of his query (entities). Only solutions at this level of advancement allow discovering the unique potential of the conversational interface, thanks to which a person can communicate with the computer "on their own terms."Today, chatbots of this type are best suited for the implementation of scenarios like "FAQs", in which the bot is able to provide answers to a large pool of questions regarding the offer of products and services of its "employer." This translates mainly to relieving the customer service office.

1.1 A Brief Introduction to Artificial Intelligence

Artificial intelligence is now a vast and progressively developing field of computer science. The number of new products, tools and services using those mechanisms have been growing rapidly. AI technologies are becoming increasingly available as practical solutions in the real world. According to Wikipedia, the term "artificial intelligence" was first used by John McCarthy in 1956 at a scientific conference in Dartmouth, where it was defined as the ability of machines to exhibit manifestations of human-like intelligence. The basis of AI are algorithms that allow computers to learn and make decisions. The most advanced programs have been inspired by the human understanding of how our brain works - the connections between neurons. One of the techniques used to build AI systems is machine learning. ML algorithms are not based on rules manually entered into the system by humans, but they automatically build models based on training data and a defined goal. Another mechanism is Deep Learning. DL models are extremely complex structures containing millions of parameters and capable of analysing complicated tasks such as human speech, images, video streams or non-trivial patterns in business data. Deep Learning models usually require a large amount of training data and computational power necessary to process them - that's why we have been observing their effects only for several years. But it is the deep models that are behind the recent successes of AI and machine learning.

1.2 Chatbot

A chatbot is a computer program used to communicate between the machine and a human. The degree of advancement such interactions can be very diverse. The simplest chatbots are simply block diagrams which, while asking more narrowly asked questions, help to reach specific answers. More complicated technologies are dialogues that detect certain words and sentences of a human being. They are able to match the closest retorts and responses. The next stage of advancement is the artificial intelligence module AI (Artificial Intelligence). Thanks to that, the computer can remember the answers we have given, as well learn and improve. The peak of advancing chatbots is the ability of software to read human speech in natural language, supported by the AI module and fluent answer to human interaction.

A chatbots are not an invention of recent years. In 1950, Alan Turing came up with the idea of creating a test as part of research on the development

of artificial intelligence. He relied on human dialogues in natural language with other people or machines. Wanting to prove the intelligence of machine, which was supposed to rely on learning abilities in the 1950s. He created the Turing test, which is still used to evaluate the effectiveness of chatbots.

I propose to consider the question, 'Can machines think?'

— Alan M. Turing
In 'Computing Machinery and Intelligence',
Mind (1950), 59, 433.

Figure 1.1: Alan Turing

The first program, which can be called a chatbot, was created in 1966 by Joseph Weizenbaum. It was named ELIZA and was quite primitive. Its operation was based on transforming the user's statements into questions, which actually gave the impression of contact with a human being, but it did not have practical application. It was necessary to wait for the creation of a breakthrough, intelligent chatbot until 1995. Then, the A.L.I.C.E (Artificial Linguistic Internet Computer Entity) program appeared, which in terms of the accuracy of the response significantly departed from its predecessors. A.L.I.C.E. was created in 1995 By Dr. Richard S. Wallace. The program was originally written in SETL - a high-level programming language based on mathematical set theory. In 1998 A.L.I.C.E. has been rewritten for Java [2]. In the 21st century, the development of chatbots progresses very fast, programs cope well with machine learning, better algorithms are created, and the systems' efficiency and server wealth allow chatbots to collect and use huge amounts of data.

A chatbot as a computer application usually consists of three parts. User is available to access the first part, a program that is a desktop or web application. The interface usually contains a window where a conversation between chatbot and user is displayed, also a field for entering text and often

a graphic representation of a chatbot. The other part of the chatbot is its engine, the heart of the program that processes the user's instructions. It implements all algorithms that allow it to recognize the entered sentences, their processing and responses. The last but the most important part of the program is its knowledge base. It contains all the phrases, sentences on which the program can operate and what it understands. This database can be any implementation and depends mainly on the programmer. This knowledge is usually stored in text files or in a relational database, which often facilitates and speeds up the search and other data operations.

Chapter 2

Context

The general context behind our project was to develop chatbot which could be trained on a given data set. Our first approach was to use Google AIY kit and modify it to our needs, however while setting it up and experimenting with it we found out we could not achieve that. There were predefined programs ready for our use, which would mean it would not require much coding from us. Another issue with that approach were the dependencies and documentation that were not properly documented.

With that in mind we have decided that the best option for us would be to start from the beginning and develop our own chatbot from scratch. The chatbot we came up with would use TensorFlow to train data on a given model classification and then build upon that to handle responses once given input from the user. Further we added client-server side so that all the necessary response data could be stored on the server side, while client took care of interaction between user and the chatbot through the use of GUI (Graphical User Interface) which was coded up using TkInter.

2.1 Project Link

Link to Repository

https://github.com/Ltrmex/Fourth-Main-Project

2.2 Goal

The aim of this project was to configure and investigate Google AIY Voice Kit from Google and configure it to our needs and purposes. Main idea behind

this was to create our own chatbot which would be capable of communication with user through graphical user interface.

Chatbot is just an idea, main goals behind this project were to familiarize ourselves with Python programming language, as it's one of the most popular ones (with Java, JavaScript and C++).

We also wanted to learn more about modernly researched technologies such as Artificial Intelligence, Machine Learning, Natural Language Processing, Neural Networks and how they can be used to create chatbot.

2.3 Objectives

2.3.1 List

1. Research possible technologies such as:

   - Chatbots.
   - Artificial Intelligence.
   - Neural Networks.
   - TensorFlow.
   - Machine Learning.
   - Python.
   - Voice Recognition.
   - Natural Language Processing.
   - Raspberry Pi and Google AIY Kit.

2. Create simple demo programs for each of the technologies and test how they function.

3. Improve on Python programming language.

4. Practice and test on Natural Language Toolkit to learn about how text can be processed.

5. Create graphical user interface.

6. Configure and test Google AIY Kit chatbot to tailor it for our purposes.

7. Integrate previously created demos into chatbot.

8. Make chatbot client-sever based.

2.3.2 Fail / Pass Criteria

Objective	Pass	Fail
1	Gained sufficient knowledge about listed technologies.	Uncertain about all, or some of the listed technologies.
2	Demos functional as intended.	Demos not functional as intended.
3	Ability to write Python code from scratch.	Inability to write code in Python.
4	Understanding of text processing.	Failed to learn how text is being processed.
5	Minimum required and functional GUI developed.	GUI unresponsive and unreliable.
6	Kit configured, altered and tested successfully.	Kit causing issues with configuration and/or alteration.
7	Prototype functioning as intended.	Prototype failed to function as intended.
8	Client-server functional and reliable.	Client-server not functional and/or unreliable.

2.4 Motivation

Our motivation to undertake this project was mainly because technologies such as Artificial Intelligence, Machine Learning, Neural Networks are being actively researched in modern industry, so we thought it would be a good practice to familiarize ourselves with those topics. We decided to work along on chatbots because we believe future is heading towards automation, where online support queries could be dealt with using a chatbot with predefined patterns to reply to clients concerns instead of human to human interaction.

2.5 Chapter Review

This paper is broken down into different chapters ranging from Methodology, Technology review, System Design, System Evaluation, Conclusion. Subsections below serves as an insight what each of the chapters is going to cover.

2.5.1 Methodology

This chapter is going to describe the way we went about our project. It's going to cover topics such as, agile development describing what approach have we used to complete this project, version control as how we managed changes in the scripts and documents, sprints to describe how development cycles went along, and testing to describe the way we have approached validation of our project.

2.5.2 Technology Review

This chapter is going to describe different technologies we have used to complete our project. Technologies included in this chapter are (but not limited to) software script, data set storage and training technologies, document typesetting system, programming languages, and many more.

2.5.3 System Design

This chapter is going to provide a detailed explanation of the overall system architecture. It's 'how' of the project. We explain various reasons for choosing certain technology over the other and how we're planning to use it in our project.

2.5.4 System Evaluation

This chapter is going to focus on evaluation of our project against the objectives set out in the introduction. In this section we're mainly trying to prove that our project is performing as intended, however, we also explaining chatbot's performance, robustness and identifying limitations.

2.5.5 Conclusion

This chapter is going to briefly try to summaries our context and objectives. It is going to remind the reader of our goals and objectives which were set out in the introduction. It also includes findings from the System Evaluation chapter, list of the outcomes of the project, and also states any opportunities identified for future investigation.

Chapter 3

Methodology

In this chapter we discuss the methodologies used in our project. A methodology is just a way to plan and control the development process of a piece of software. There are various methodologies to choose from including Extreme Programming, Rapid Application Development or Waterfall but for this project we used Agile as our main methodology.

3.1   Agile Development

During development of this project, a group took an Agile approach, which is far more convenient as opposed to the more traditional approach such as the Waterfall. As such, the project was developed using iterative methods that adapted to any changes in the requirement, or the development process.

During the research and development process of our project we used a Scrum like approach. Scrum is an Agile method in which a development cycle is carried out in what are known as sprints which are described in Section 3.3.

Throughout the life cycle of the project we held weekly meetings to discuss what had to be completed next. Upon coming to a decision we then would meet with our supervisor to discuss whenever our approach was correct. Upon agreement with our supervisor then we would split the work evenly.

3.2   Version Control

Throughout the life cycle of our project we used GitHub. GitHub is a hosting service for version control. We created a GitHub repository, to work as

collaborators. We found GitHub to be an extremely useful tool in the research and development of our chatbot.

Work that was completed was stored locally on the machine. In order for the work to be contributed, it had to be uploaded to GitHub repository. Therefore, initially, we needed to clone the online repository. Cloning would be done through command prompt which is a command line interpreter(cmd for short), using 'git clone' command which would copy the online contents of the repository to the current local directory.

- By using git pull command we could fetch and merge changes on the remote server to the local working directory. This made sure that the latest version of the project and its contents is stored locally

- Then git commit -m description of the commit command is used to commit changes locally and prepare them for merging event with the online repository.

- Git push command is used to send or push changes to the master branch of the remote repository.

3.3 Sprints

3.3.1 Basic Chatbot

This was a relatively small sprint and was completed by Maciej. The plan for this sprint was to develop a simple chatbot which would be able to mimic and make connections with whatever user typed into command prompt. The connections were made by the use of SQLite where table contained word, sentence and association, and that way chatbot could respond in a-like smart way after certain amount of training.

3.3.2 Neural Network

As this sprint was a huge part of our project, we decided to both work on it. Kamila was responsible for research and investigation of it. While Maciej was responsible for creating a simple neural network, training it and testing it. At the end of this sprint we managed to produce a simple neural network and we managed to learn how they work and how they can be used in our application.

3.3.3 Speech Recognition

This was a relatively small sprint and was completed by Kamila. The plan for this sprint was to create set of demo programs which would be responsible for speech recognition:

- Speech to text - converting speech and displaying what was said in text.

- Text to speech - typing in command prompt a sentence which ten would be spoken.

3.3.4 Natural Language Processing

This phase of the project took us some time as we were discovering how Natural Language Processing functions and what's it's use. Main goal of this sprint was to discover how could we use NLP in our project for text and data processing. It took us few weeks to investigate and test various functions from Natural Language Toolkit package. Upon completion of this sprint we familiarized ourselves with functions such as tokenizing, stemming, chinking lemmatizing, but also discovered how to access and use corporas data set and make classifiers, which later could be used for training data etc.

3.3.5 Google AIY Kit

This sprint we treated like an experiment. We wanted to get familiar with AIY Voice Kit from Google, its structure and find out how to use the Speech-to-Text service works with Google Assistant. However, before using the Voice Kit we had to connect all the hardware that was provided to us. It was a challenging experience for us because we never used Raspberry Pi before. The next step was to create a project on the Google Cloud Platform thanks to which we could turn on the Google Assistant API.

3.3.6 Graphical User Interface

Next phase of the project required us to familiarize ourselves with how graphical user interfaces could be integrated into Python programming language. There were few ways to achieve this, given variety of GUI toolkits available online. We have decided to use Tkinter for our GUI development as it was included as a standard in the Python library. There was another choice to use GuiZero, however it wasn't reliable.

3.3.7 Client-Server

This sprint was completed by Maciej. It required to develop a client-server code to ease the use of the chatbot. Server developed is multi-threaded which means it allows up to five users to interact, in the same chat, with the chatbot. On the server side is where all the training data, models, and chatbot logic is stored, making it easier on the client side. On the client side, once server has been launched user can communicate with chatbot through the use of sockets in the background.

3.3.8 Model-Response Chatbot

This phase was crucial for our project because on its basis we started building a chatbot. The goal of this sprint was building a chatbot framework using TensorFlow and add some context handling. The first step was the structure in in conversions intents were defined. Then we had to organize our documents, words and classification classes. Next, we built and trained our model using the TFLearn deep learning library. Another goal of this sprint was adding a class that will be responsible for chatbot response. It aims to load our saved TensorFlow model and classified each sentence passed to response.

3.3.9 Data Filtering

This phase of the project was important for our project as it was responsible for generating data set for training. It has three states:

- Getting the data.

- Converting from text file to comma-separated values(csv, ie excel format) format.

- Csv to JavaScript Object Notation(JSON) format.

These steps were necessary as data that our chatbot was trained on originally was in JSON format and we did not want to complicate it any further. The data was extracted from Corpora data set, which is just a collection of different conversations.

3.4 Testing

During the testing and validation phase of our project we decided not to use any of the frameworks like Junit. However we decided to use both white box

testing and black box testing.

3.4.1 White Box Testing

White box testing[3] is a software testing method in which the internal structure/design/implementation of the item being tested is known to the tester. The tester chooses inputs to exercise paths through the code and determines the appropriate outputs. Programming know-how and the implementation knowledge is essential. White box testing is testing beyond the user interface and into the analysis of the internal structure of a system. This method is named so because the software program, in the eyes of the tester, is like a white/transparent box; inside which one clearly sees.
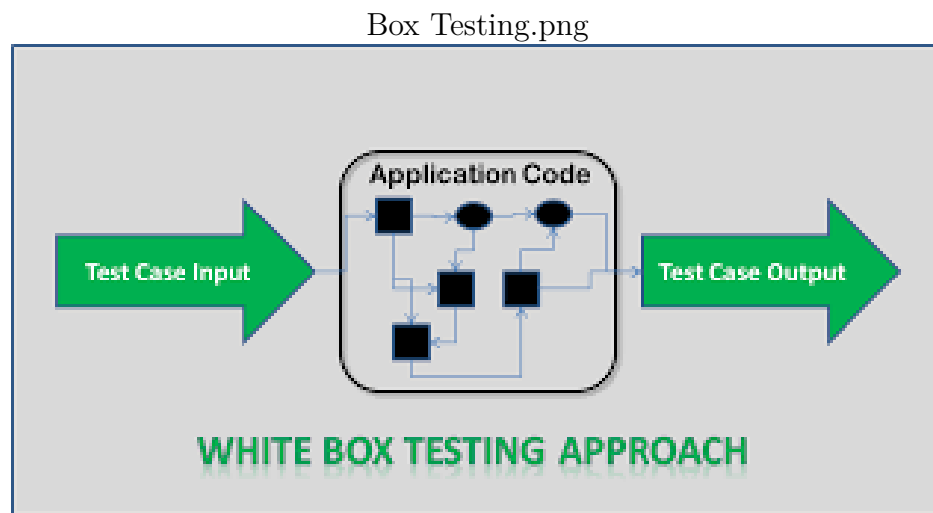
Box Testing.png

Figure 3.1: White Box Testing

The way we used white box testing was one of us developing a certain feature and then the other would test that feature's functionality. One doing the testing would present input, follow the path through the code and then examine the output.

3.4.2 Black Box Testing

Black box testing[4] also known as Behavioral Testing, is a software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional. This method is named so because

the software program, in the eyes of the tester, is like a black box; inside which one cannot see. This method attempts to find errors in the following categories:

- Incorrect or missing functions.

- Interface errors.

- Errors in data structures or external database access.

- Behavior or performance errors.

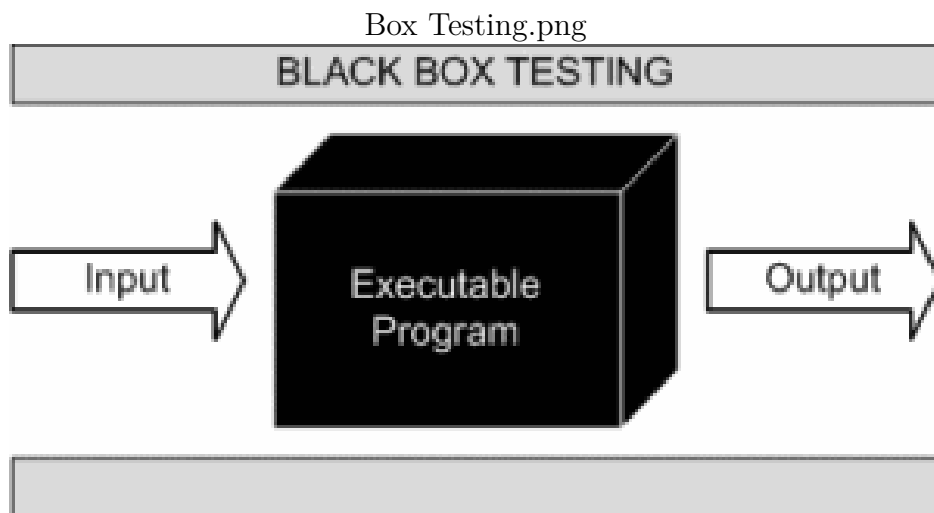- Initialization and termination errors.

Box Testing.png

Figure 3.2: Black Box Testing

The way we used black box testing was asking our friends to use our chatbot features, for example, using our graphical user interface and then reporting their experience. This kind of testing was useful as it gave us an insight of how users would use our chatbot and how it was doing performance wise.

Chapter 4

Technology Review

In this section we will discuss the various technologies that were used in the development of our project.

4.1 Visual Studio Code

Figure 4.1: Visual Studio Code logo

Visual Studio Code is a free and modern programming editor developed by Microsoft for users of Windows, Linux and OS X. The software has support for debugging code, managing source code versions via Git version control system, automatic IntelliSense code completion, code block management and refactoring. The integrated debugger works both at the source code level, and the machine level. The program can be used for writing, among others, in JavaScript, TypeScript, C++, C, HTML/CSS and much more. Other tools within Visual Studio are a designer for creating Windows Forms, Windows

Presentation Foundation (WPF) and web applications, a tool for creating classes, designing databases, etc. The functionality of Visual Studio can be extended at any level of the application using add-ons.

We chose Visual Studio Code as a code writing tool because it has a well-structured interface and the user does not need to have much experience with programming to use it. Another great feature is IntelliSense, which suggests the syntax of the code. Also uses guides that add helper lines to the code. Thanks to this, it is much easier to grasp the indentation code. However biggest advantage is git support. The Source Control is a that can track all the changes made in the repository. Using Source Control Providers extensions, you can easily push our changes to GitHub.

4.2   GitHub

Figure 4.2: GitHub logo

GitHub it's a web hosting service for programming projects using the Git version control system. GitHub was developed by Chris Wanstrath, P.J. Hyett, Tom Preston-Werner and Scott Chacon using Ruby on Rails. GitHub provides free hosting of open source programs and paid private repositories. While Git is a command line tool, it provides a Web-based graphical interface. It's a great social networking site for developers and programmers. GIT creates a story for the application folder and allows to manage its versions and work for many people on one project. Each progression that was made is pushed to the repository and is called a" commit". Developers can check individual commits and track the progress as well as see any deletions/additions to the existing code.

As the version control system, we chose GitHub because it facilitates working in a group, where everyone can work on their copy of the project without worrying about spoiling the work of someone else. The documentation is widely available online. There are a lot of different ways to use Git. There are the original command-line tools, and there are many graphical user interfaces of varying capabilities.

4.3 Python

Figure 4.3: Python logo

Python is a high-level programming language with a general purpose. The design philosophy puts emphasis on programmer productivity and code readability. It has a minimalist core syntax with few basic commands and simple semantics, but it also has large and comprehensive standard libraries, including an Application Programming Interface (API) with many basic operating system (OS) functions. Python code, by virtue of minimalism, defines built-in objects such as list, tuples, dictionaries (dict) and arbitrarily long numbers (long). Python supports many programming paradigms, including object-oriented programming (class), procedural programming (def) and functional programming (lambda). Python has a dynamic type system and automatic memory management with reference counting (like Perl, Ruby and Scheme). This programming language was created in the early 1990s (1991) by the Dutch programmer Guido van Rossum at CWI (Centrum voor Wiskunde en Informatica). Although a large contribution to the creation of Python is also attributed to other people, however, Van Rossum is its main creator. Contrary to appearances, the name of the language is not associated with the animal but with the series broadcast by the BBC in the seventies, of

which Van Rossum was a fan. Applications written in Python work under many systems such as Windows, Linux / Unix, Mac OS X, OS / 2, Amiga, and Palmphones and Nokia smartphones. There are also Python implementations in Java (Jython) and .NET (IronPython) that work wherever these platforms are available. Python can be used to create websites, desktop applications running on users' computers, including games, in web applications or scripts, e.g. generating statements and reports. In addition, Python is distributed on an open license, which allows it to be used also for commercial projects. It is a versatile programming language.

We chose python as a programming language because Easy to learn, it has a simple, almost intuitive syntax. Keywords are therefore understandable for every beginner, but at the same time very close to those occurring in other programming languages. The important aspect was that Python provides generators both as expressions and functions. The generators allow iterative data processing - element by element. The generators allow you to retrieve data from the source one element and pass them through the entire processing chain bypassing the mechanism associated with storing the iterated list. Python works great for machine learning and artificial intelligence. Python's huge capabilities have been appreciated by giants like NASA, Dropbox, YouTube and Reddit. Python is also used by Spotify and Netflix. According to Application Developing Trends[5] in 2018, Python was Language of the Year which also confirmed our choice of language.

4.4 JSON

Figure 4.4: JSON logo

JSON (JavaScript Object Notation) is a text-based way of writing data independent of the computer platform. It is an open-standard file format and is used to transfer data that can be included in tables in objects. JSON represents any data structure (number, string, boolean, null, object or a complex array of them) stored in text form (string). Douglas Crockford first specified [6] and popularized the JSON format. The acronym originated at State Software, and the company co-founded by Crockford and others in March 2001. JSON, like other structures used to store data such as XML, is widely used. Most often, however, it is used to transmit and receive data from the server through applications on the website. For JSON to be valid, the value keys must be strings surrounded by double quotes. A single quotation mark will not work in this case! Another thing to watch out for is adding a comma after the last object.

In our project, we decided to use JSON format because it is an easier to use alternative to XML. Specification is short and well explained. JSON is easier to read than XML. The files are more restrictive. JSON can be analyzed using a standard JavaScript function. The JSON data structure is less complex and full because JSON limits the programmer to the objects to be modeled. This makes the code easier to read and more predictable.

4.5 Google Cloud Platform

Figure 4.5: Google Cloud Platform logo

Google Cloud Platform (GCP) it's a platform that offers cloud-based services. Google uses the same infrastructure to provide customers with their own

products[7]. Initial release of, Google App Engine was in April 2008, Google-managed data canters. However, the service was available in November 2011. Google Cloud Platform (GCP) offers dozens of IaaS, PaaS and SaaS services. Cloud Platform, or services provided in the cloud computing model, currently includes:

- Google App Engine - a universal environment that allows application developers to focus on creating applications in one of the popular programming languages (Java, Python, GO, PHP), leaving the issue of platform maintenance and security hardware and software for Google;

- Google Cloud Storage - a solution for storing large data sets;

- Google Cloud SQL - a solution enabling the use of SQL databases in a cloud model;

- Google Cloud Compute - competitive solution to AWS, allowing you to run your own virtual machines on the Google platform;

- Google Big Query - a solution for processing large data sets (Big-Data).

In order to use Google Assistant and Cloud Speech APIs with our Voice Kit, we had to create a project on the Google Cloud Platform as well as get credentials from Google's developer console.

4.6   AIY Voice Kit from Google

Figure 4.6: AIY Voice Kit from Google

AIY Projects is a series of open-source designs that demonstrate how easy it is to add AI to projects. The AIY Voice Kit [8] from Google allows to build a cardboard device that uses the Google Assistant to answers questions and issue voice commands. It helps to broaden knowledge about voice recognition and natural language understanding. Instructions contains information and explanation how to add voice command to the project. In the kit, Google has included a microphone, a speaker, and an accessory board called Voice Hat, that is loaded with breakout pins to wire up a variety of sensors and components.

4.7 Natural Language Processing
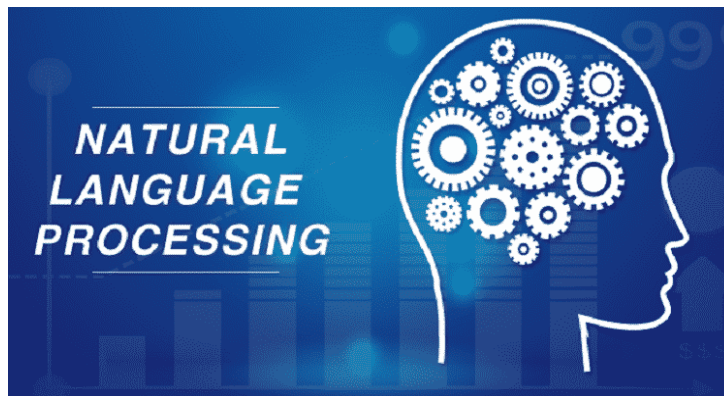
Figure 4.7: Natural Language Processing

Natural language processing (NLP) [9] - is a sub-field which combines the issues of artificial intelligence and linguistics, dealing with the automation of analysis, understanding, translation and generation of natural language by a computer. The natural language generating system transforms the information stored in the computer's database into an easy-to-read and understandable language. And a system that understands natural language converts natural language samples into more formal symbols, easier to process for computer programs. Many NLP problems are related to both generation and understanding of the language, for example, the morphological model of the sentence (structure of words), which the computer should build, is also needed to make the sentence understandable and grammatically correct. NLP also coincides to a large extent with the department of computer linguistics and is often regarded as a subdivision of artificial intelligence. By contrast, the term natural language is used to distinguish between human

languages (such as Spanish, Swedish or Polish) from a formal or computer language (such as C ++, Java or Lisp). Although the analysis of natural language can deal with both text and speech, work on the synthesis of speech has evolved as a separate department. Natural language processing using techniques that were very useful in our project such as:

- Tokenizer – splitting the content into words or expressions.

- Stemming and lemmatization - normalizing words so unique structures guide to the sanctioned word with the same meaning. For instance, "running" and "ran" guide to "run."

- Entity extraction - distinguishing subjects in the content.

- Part of speech detection - distinguishing content as a verb, noun, participle, verb phrase, and so on.

- Sentence boundary detection - detecting complete sentences within paragraphs of text.

4.8   TensorFlow

Figure 4.8: TensorFlow logo

TensorFlow is a set of open-source software libraries for the development of machine learning. They have been specially designed for the construction and training of neural networks. They allow to detect and create patterns and correlations analogous to human learning and reasoning. TensorFlow libraries were developed by the Google Brain team, initially for personal use only. Their first version was released on November 9, 2015 and was

released to the public on February 11, 2017. The latest update, marked 1.3.0, was published on August 17. By browsing the contents of tensorflow.org [10], we see that Google has provided not only the library itself, but also neural network models, trained to recognize photos and handwriting, analyse text and speech. It's enough to push forward deep learning algorithms, but not only - TensorFlow is matched to other forms of Artificial Intelligence, including logistic regression and reinforcement learning. It supports Linux, mac-OS, Windows, there is also a version mobile iOS and Android.

In our project, we used TensorFlow because the Dataflow model enables very easy and effective parallelization of operations. The model was created in Python, which we use as our programming language in the project. TensorFlow has many options for deep learning (Deep Learning) - it optimally facilitates the creation of a neural network, its use and learning.

4.9   TFLearn

TFLearn is high level abstract API for TenserFlow. It's a morular and transparent deep learning library build on top of TensorFow. It was designed to provide a higher-level API to TensorFlow in order to facilitate and speed-up experimentations, while remaining fully transparent and compatible with it. TFLearn offers a quick way for Data Engineers or Data Scientist to start building Tensorflow neural networks without having to go deep into Tensorflow. Neural Networks with TFLearn are still written in Python, but the code is drastically reduced from Python Tensorflow. Using TFLearn provides Data Engineers new to Tensorflow an easy way start learning and building their Deep Neural Networks (DNN).

TFLearn was used in our project because it provides a high-level API for implementing deep neural networks very good documentation. Code samples are available and described in detail for the user who has not previously had experience with another TensorFlow library. It allows to quickly change from writing TFLearn code to TensorFlow code without additional problems. TFLearn provides a wrapper called Deep Neural Network (DNN) which we used in our project that automatically performs neural network classifier tasks, such as training, prediction, save/restore, and more.

4.10 SQLite3

Figure 4.9: SQLite3 logo

SQLite is a software library that implements stand-alone and server-less, non-demanding SQL database engine configuration. It was created by Richard Hipp and is available under the public domain license. The project was started in 2000[11]. The SQLite database is currently very popular. SQLite is used on Solaris 10 and MAC OS, iPhone and Skype. The SQLite engine does not create an independent process for the database. Instead, it is statically or dynamically attached to the application. The SQLite library has a small size of approximately 470 kB. The SQLite database is a single file on the disk that can be placed anywhere in the directory hierarchy. It can be used on various operating systems, both 32 and 64 bits. SQLite was written in the programming language C and has bindings for many languages like C ++, Java, C , Python, Perl, Ruby, Visual Basic and others. The main advantage of this database is performance especially when performing INSERT and SELECT type queries, as well as cross-platform. Unfortunately, this is not a solution free of defects. During the process of adding new data, SQLite must block the entire file until the operation is completed. Therefore, this solution is not efficient in situations where data is constantly changing. The SQLite library contains a simple command line tool called sqlite3 (sqlite3.exe for Windows systems), which allows the user to manually enter and execute SQL commands on the SQLite database.

Our choice fell on SQLite in our project with all its advantages. It works on files, which means that the whole database consists of one file on the disk, which allows you to easily move the database. It is ideal for design and even testing. Reading and writing from an SQLite database is often faster

than reading and writing individual files from disk. Content can be accessed and updated using concise SQL queries instead of lengthy and error-prone procedural routines.

4.11 MySQL

Figure 4.10: MySQL logo

MySQL is a fast and reliable relational database management system (RDBMS). Databases allow for efficient storage, retrieval, sorting and obtaining information. It is also available under the Open Source license, but if necessary, you can purchase its commercial version. The MySQL server controls access to data, makes them available to many users at the same time, and ensures the fastest possible use of them. In addition, the server controls user authentication, which allows the data to be used only by authorized persons. It is a multi-user and multi-threaded system that uses the SQL standard, which makes it possible to query the database. MySQL was created by a Swedish company MySQL AB in 1995 [12]. The developers of the platform were Michael Widenius (Monty), David Axmark and Allan Larsson.

In our opinion it is worth using MySql because it is very easy to install it. Visual (GUI) make it extremely simple to start working with a database. It has many security features, some quite advanced, are built into MySQL.

4.12 Tkinter

Figure 4.11: Tkinter logo

TkInter is (in the sense of Python semantics) a module that provides a set of functions, constants and objects. Tkinter is a library that allows to create a graphical interface for Python, based on the Tk library. In Tkinter, the IDLE editor has been created as a standard part of Python. A GUI program using TkInter usually consists of the following four elements: importing the tkinter module, creating the main program window, adding necessary widgets to the window and launching the event controller. Tk is used by various languages (Perl, Ruby, PHP, etc.). Occur on various platforms (Unix/Windows /MacOS). Tkinter provides various controls, such as buttons, labels and text boxes. These controls are commonly called widgets. There are currently 15 types of widgets in Tkinter.

4.13 CSV

Figure 4.12: CSV logo

Content in CSV format refers to data files attached to the .csv extension, and these CSV files are also called comma separated values. Comma-separated values is a data format that pre-dates personal computers by more than a decade: the IBM Fortran (level H extended) compiler under OS/360 supported them in 1972 [13]. The "comma-separated value" name and "CSV"

abbreviation were in use by 1983 [14]."CSV" in a file placed with the .csv extension means "comma-separated values" because the data from these CSV files are comma-separated details for the individual information sets. These data elements can be entered by users of spreadsheets and text editing applications integrated to support the creation and modification of CSV documents. Separate rows of the database are represented by each line of text saved in the CSV file. These database rows are implemented from the field one or more data, and these are separated by commas. Microsoft Windows users can install spreadsheets and text editing applications developed by Microsoft to store data that can be later used in database applications. One of those programs that can be used to create, open, view and edit the contents of a CSV file is a Microsoft Excel spreadsheet. There are also word processors and spreadsheet that can be installed and used by Linux and Mac users to implement support for creating, opening, viewing and modifying the contents of these CSV files.

We decided to use CSV because its great advantage is the universal character, everything can be store in it that can be saved in the form of text (characters, strings, numbers) and use the form in the form of a table (column name and value) . Many applications, such as online stores, use this format to exchange information with the user but also among other services.

4.14   LaTex

Figure 4.13: LaTex logo

Latex is used to produce clear-looking text documents such as books, articles or even presentations. The target format is a printout, or files in various

formats, such as PDF, Postscript or HTML. It is particularly convenient to create technical and mathematical documents, but it can also be successfully used to write program documentation or a collection of stories. LaTeX was originally written in the early 1980s by Leslie Lamport. Latex, like programming languages, has its own language in which the document's content is written and has tools (you could say "compilers") that process source files and generate target files. In programming languages, one of the most important things is the collection of libraries with ready implementations of various typical activities. Also, in Latex there are many ready-made packages that allow you to quickly create various elements and types of documents.

Latex's philosophy is to focus on what the content of the document should contain, and how little to devote attention to how it should look. In other words, we only introduce the structure and content of the document, and latex does the rest of the work for us to make the original document look like it should. Of course, we have a great opportunity to interfere with the look, but usually it's just choosing a template or the need to get a non-standard effect. It is a completely different philosophy than in many other text editors, especially in various office applications, where we must decide at almost every step what the appearance, size, font, spacing, display of titles, etc. should be.

Chapter 5

System Design

5.1   Architecture
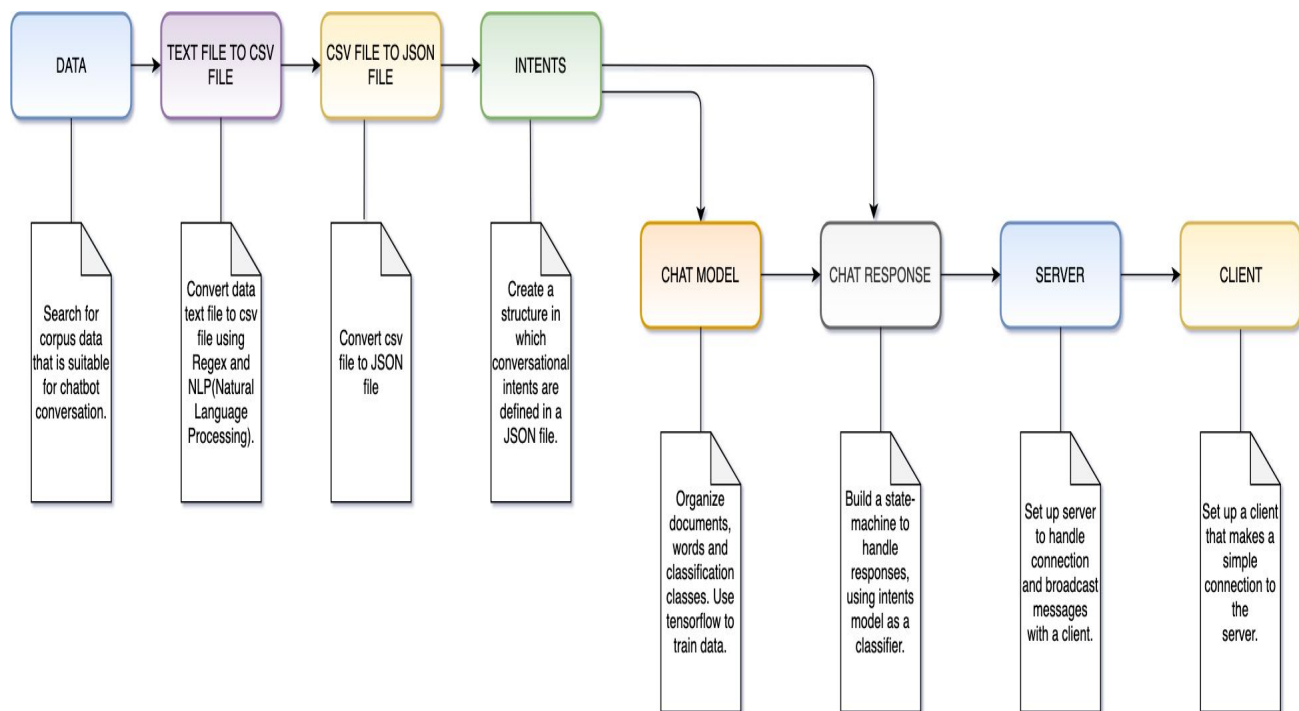
Figure 5.1: Chatbot Architecture

5.2   Data Design

A data set[15](or dataset) is a collection of data. Most commonly a data set corresponds to the contents of a single database table, or a single statistical data matrix, where every column of the table represents a particular variable, and each row corresponds to a given member of the data set in question. The data set lists values for each of the variables, such as height and weight of an object, for each member of the data set. Each value is known as a datum. The data set may comprise data for one or more members, corresponding to the number of rows.

The term data set may also be used more loosely, to refer to the data in a collection of closely related tables, corresponding to a particular experiment or event. Less used names for this kind of data sets are data corpus and data stock.

5.2.1   Data extraction

For data extraction we have used an early release of the self-dialogue Corpus containing 24,165 conversations, or 3,653,313 words, across 23 topics, those are stored inside corpus folder. Those conversations can be extracted in various ways depending on arguments parsed. The results are then stored in dialogues folder. Inside the python program file responsible for extraction of that data we have a few arguments that can be passed in:

- inDir: directory to read corpus from.

- outDir: directory to write processed files.

- output-naming: whether to name output files with integers (integer) or by assignment id (assignment id).

- remove-punctuation: removes punctuation from the output.

- set-case: sets case of output to original, upper or lower.

- exclude-topic excludes any of the topics (or sub-directories of corpus), e.g. –exclude-topic music.

- include-only includes only the given topics, e.g. –include-only music.

  Methods used in this part of the project are as follow:

- write-dialogues(): Writes dialogues given keys to directory.

- read-data(): Reads data from CSVs in a directory and returns a dictionary indexed by assignment IDs.

- create-dirs(): Creates training directories. Returns their paths.

- parse-args(): Parse input arguments which were mentioned above.

5.2.2 Data conversion

Data conversion[16] is the conversion of computer data from one format to another. Throughout a computer environment, data is encoded in a variety of ways. For example, computer hardware is built on the basis of certain standards, which requires that data contains, for example, parity bit checks.

Data conversion works in two ways in our project. First we loop through all the conversations that were generated (and are stored in dialogues folder), it tokenize sentences which are then stored as responses. Then it converts from the text format to single file in comma separated values format. Second step in this is conversion from previously generated CSV file into JavaScript Object Notation format. Main reason for this conversion is that chatbot model was set up in a way that it only accepts JSON file formats for its model training.

5.2.3 Clean up

Sometimes while converting from one format to another, some issues may arise, so its critical that after successful extraction and conversion of data, that the data is reviewed. Appropriate way to do this is to check if the file is correctly set up and test it out. If the conversion went well then data can be added to the main data set which is stored inside intents.json.

In our case there was some clean up that needed to be done. Main reason for this was because JSON file wasn't structured properly for our purposes. In our case we needed to use Notepad++ to replace certain structures inside JSON file, those changes have been recorded inside convert.txt file which is available on GitHub.

5.2.4 Model Training and Response Handler

This phase was crucial for our project because on its basis we started building a chatbot. During model training we have building a chatbot framework

using TensorFlow and added some context handling. The first step was the structure in in conversions intents were defined. Then we had to organize our documents, words and classification classes. Next, we built and trained our model using the TFLearn deep learning library.

As this phase of our project was one of the crucial ones it will be explained in more detail. The steps to complete model training were as follows:

Our chatbot framework needed a structure in which conversational exchanges could be stored. Clean way to store our data was through the use of JavaScript Object Notation, for example:

```
{"intents": [
        {"tag": "mopeds",
         "patterns": ["Which mopeds do you have?",
         "What kinds of mopeds are there?",
         "What do you rent?" ],
         "responses": ["We rent Yamaha, Piaggio and Vespa mopeds",
         "We have Piaggio, Vespa and Yamaha mopeds"]
        },
        {"tag": "payments",
         "patterns": ["Do you take credit cards?",
         "Do you accept Mastercard?", "Are you cash only?" ],
         "responses": ["We accept VISA, Mastercard and AMEX",
         "We accept most major credit cards"]
        }
]}
```

As you can see each conversational intent contains a tag for a unique name, patterns which are sentence patterns for our neural network text classifier and responses which one out of many will be used as a response.

There were a few imports also that were required which are:

```
import nltk
from nltk.stem.lancaster import LancasterStemmer
stemmer = LancasterStemmer()

# Imports for Tensorflow
import numpy as np
import tflearn
import tensorflow as tf
import random
```

```python
# Other imports
import pickle
import json
```

With the data set file in JSON format loaded, next step was organize the documents, words and classification classes. We created a list of documents (sentences), each sentence is a list of stemmed words and each document is associated with an intent (a class). The stem 'tak' will match 'take', 'taking', 'takers', etc. We could clean the words list and remove useless entries but this will suffice for now:

```python
words = []
classes = []
documents = []
ignore_words = ['?']
# loop through each sentence in our intents patterns
for intent in intents['intents']:
    for pattern in intent['patterns']:
        # tokenize each word in the sentence
        w = nltk.word_tokenize(pattern)
        # add to our words list
        words.extend(w)
        # add to documents in our corpus
        documents.append((w, intent['tag']))
        # add to our classes list
        if intent['tag'] not in classes:
            classes.append(intent['tag'])

# stem and lower each word and remove duplicates
words = [stemmer.stem(w.lower()) for w in words if w not in
ignore_words]
words = sorted(list(set(words)))

# remove duplicates
classes = sorted(list(set(classes)))

# create a list of documents (sentences), each sentence is a list
# of stemmed words and each document is associated with an
# intent (a class).
print(len(documents), "documents")
```

```
print (len(classes), "classes", classes)
print (len(words), "unique_stemmed_words", words)
```

Unfortunately that data structure would not work with Tensorflow, we needed to transform it further, from documents of words into tensors of numbers. Our data is being shuffled. Tensorflow takes some of this and use it as test data to gauge accuracy for a newly fitted model. If we look at a single x and y list element, we see 'bag of words' arrays, one for the intent pattern, the other for the intent class.

```
# create our training data
training = []
output = []
# create an empty array for our output
output_empty = [0] * len(classes)

# training set, bag of words for each sentence
for doc in documents:
    # initialize our bag of words
    bag = []
    # list of tokenized words for the pattern
    pattern_words = doc[0]
    # stem each word
    pattern_words = [stemmer.stem(word.lower()) for word in
    pattern_words]
    # create our bag of words array
    for w in words:
        bag.append(1) if w in pattern_words else bag.append(0)

    # output is a '0' for each tag and '1' for current tag
    output_row = list(output_empty)
    output_row[classes.index(doc[1])] = 1

    training.append([bag, output_row])

# shuffle our features and turn into np.array
random.shuffle(training)
training = np.array(training)

# create train and test lists
train_x = list(training[:,0])
```

```
train_y = list(training[:,1])
```

Next step was to build our model.

```
# Build a model
# reset underlying graph data
tf.reset_default_graph()
# Build neural network
net = tflearn.input_data(shape=[None, len(train_x[0])])
net = tflearn.fully_connected(net, 8)
net = tflearn.fully_connected(net, 8)
net = tflearn.fully_connected(net, len(train_y[0]),
activation='softmax')
net = tflearn.regression(net)

# Define model and setup tensorboard
model = tflearn.DNN(net, tensorboard_dir='tflearn_logs')
# Start training (apply gradient descent algorithm)
model.fit(train_x, train_y, n_epoch=1000, batch_size=8,
show_metric=True)
model.save('model.tflearn')
```

Further step was to build a simple state-machine to handle responses, which would use previously created intents model as a classifier. Previously mentioned imports need to be included again at the top of the program. Next step then is to un-pickle previously saved model, as well as reload the intents file. While chatbot framework is seperate from model build, the model needs to be rebuilt if the intent patterns change.

Next step was to load previously saved trlearn framework, as well as define the TensorFlow structure as we did before. Before processing intents it was crucial to produce bag of words from user input in a familliar way we have done above.

```
def clean_up_sentence(sentence):
    # tokenize the pattern
    sentence_words = nltk.word_tokenize(sentence)
    # stem each word
    sentence_words = [stemmer.stem(word.lower()) for word in
    sentence_words]
    return sentence_words
```

```
# return bag of words array: 0 or 1 for each word in the bag that
# exists in the sentence
def bow(sentence, words, show_details=False):
    # tokenize the pattern
    sentence_words = clean_up_sentence(sentence)
    # bag of words
    bag = [0]*len(words)
    for s in sentence_words:
        for i,w in enumerate(words):
            if w == s:
                bag[i] = 1
                if show_details:
                    print ("found in bag: %s" % w)

return(np.array(bag))
```

Next step was to build response processor. Each sentence which was passed in to response function is classified. Classifier uses model.predict(). The probabilities returned by the model are lined-up with our intents definitions to produce a list of potential responses. If one or more classification are above a threshold, we see if a tag matches and intent and then process that

Each sentence passed to response() is classified. Our classifier uses model.predict() and is lighting fast. The probabilities returned by the model are lined-up with our intents definitions to produce a list of potential responses. Classification list was treated as a stack which would pop off the stack looking for a suitable match until we find one, or it's empty.

```
ERROR_THRESHOLD = 0.25
def classify(sentence):
    # generate probabilities from the model
    results = model.predict([bow(sentence, words)])[0]
    # filter out predictions below a threshold
    results = [[i,r] for i,r in enumerate(results)
    if r>ERROR_THRESHOLD]
    # sort by strength of probability
    results.sort(key=lambda x: x[1], reverse=True)
    return_list = []
    for r in results:
        return_list.append((classes[r[0]], r[1]))
    # return tuple of intent and probability
```

```
    return return_list

def response(sentence, userID='123', show_details=False):
    results = classify(sentence)
    # if we have a classification then find the matching
    intent tag
    if results:
        # loop as long as there are matches to process
        while results:
            for i in intents['intents']:
                # find a tag matching the first result
                if i['tag'] == results[0][0]:
                    # set context for this intent if necessary
                    if 'context_set' in i:
                        if show_details: print ('context:', i['context
                        context[userID] = i['context_set']

                    # check if this intent is contextual and
                    # applies to this user's conversation
                    if not 'context_filter' in i or \
                        (userID in context and 'context_filter'
                        in i and i['context_filter']
                        == context[userID]):
                        if show_details: print ('tag:', i['tag'])
                        # a random response from the intent
                        return random.choice(i['responses'])

            results.pop(0)
```

5.2.5 Server Client

A server is a computer program or a device that provides functionality for
other programs or devices, called "clients". This architecture is called the
client–server model, and a single overall computation is distributed across
multiple processes or devices. Servers can provide various functionalities,
often called "services", such as sharing data or resources among multiple
clients, or performing computation for a client. A single server can serve
multiple clients, and a single client can use multiple servers. A client process
may run on the same device or may connect over a network to a server on a
different device.

For our server-client connection we decided to use TCP sockets over UDP sockets because they're more telephonic, where the recipient has to approve the incoming connection before communication begins. UDP sockets are more post-mail which means that anyone can send to any recipient that is known so they don't really require an establishment of connection before communication can happen. TCP suit more to our purpose than UDP sockets, therefore we use them.

The way our server works is that it waits for incoming connections and as soon as it gets one, it logs the connection (prints some of the connection details) and sends the connected client a welcome message. Then it stores the client's address in the addresses dictionary and later starts the handling thread for that client. Where the server allows up to five connections at the same time to communicate with the chatbot in the same chat. On server side is also where the chatbot logic is stored (chatbot response) so server is responsible for parsing users message to the chat-response.py which then returns appropriate response. Server then broadcasts parsed input and response onto all the clients that are connected to the server.

Client is then responsible for handling responses from the server and sending messages to server to handle.It also uses socket connection as mentioned above. When launching the client it will ask for host IP as well as a port it should connect to. Its critical to parse in proper information otherwise the connection will be denied.

Another feature that client is responsible for is its graphical user interface. We have decided to create simple easy for eyes GUI where everything is self explanatory. GUI was created through the use of TkInter which is is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit, and is Python's de facto standard GUI and is included in most operating systems as a default.

5.3 Google AIY Voice Kit Design

Figure 5.2: Google AIY Voice Kit

AIY Voice Kit was the first project from Google that lets user explore voice recognition and natural language understanding. Our task was to build a cardboard device that uses the Google Assistant to answer questions like "what is the weather in Ireland?" and understand given commands.

5.3.1 Getting to know the hardware
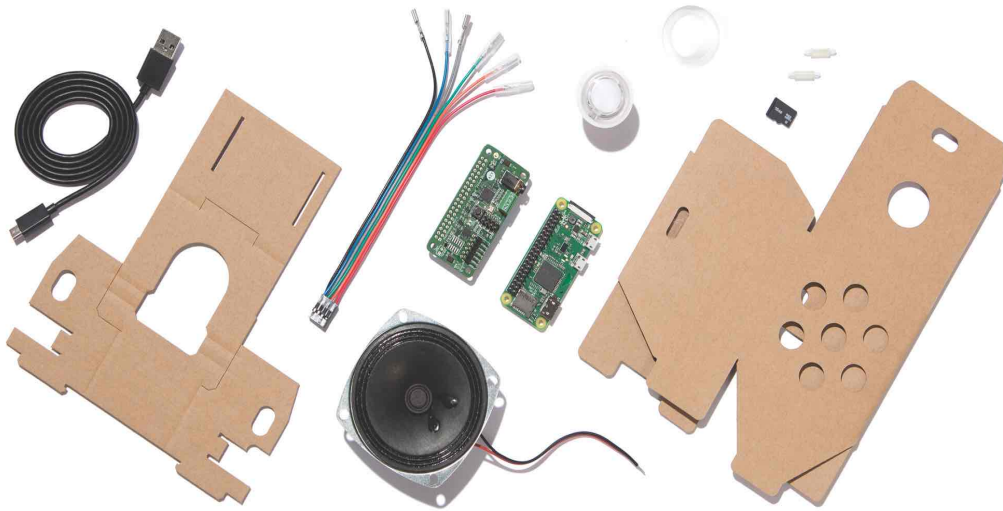
Figure 5.3: List of materials used to build a Voice Kit

Our kit included set of components to build a voice-capable device with Raspberry Pi which you can see in the picture above. A HAT board is needed to connect all accessories together; the other is a stereo microphone. Using these components along with provided instruction we were able to build our first AIY Project kit.

5.3.2 Setting up the software and getting the latest system image

Figure 5.4: Software settings

First step was to download the AIY Project image from magpi. Next, we had to burn the image to our microSD card using program Etcher [17]. Respectively we inserted SD card into the slot of the Raspberry Pi board. With the microSD card inserted and the peripherals connected we were able to see the AIY Project desktop.

5.3.3 Building a Voice Recognizer
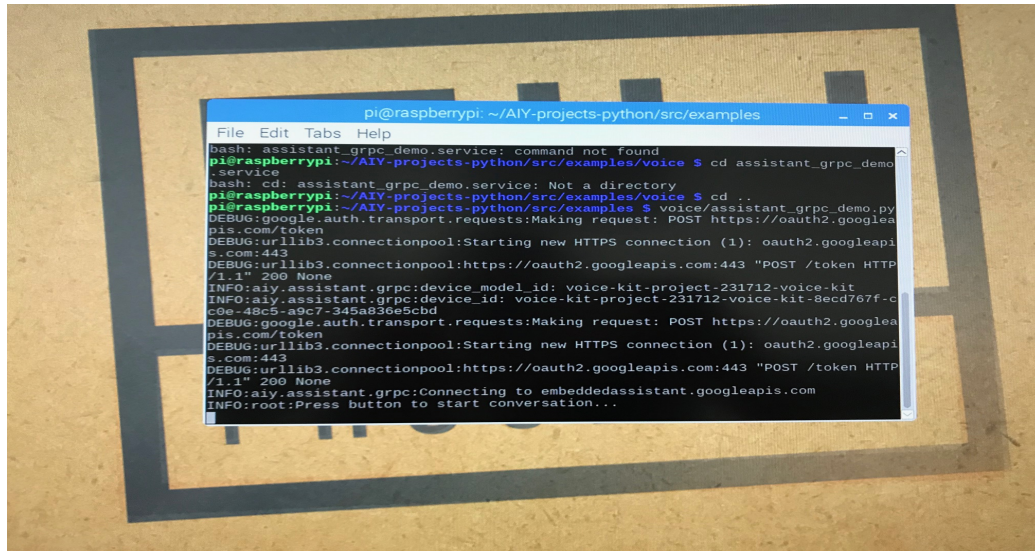
Figure 5.5: Voice Recognizer

The purpose of this phase was to build a voice recognizer that can use Google Assistant SDK [18] to recognise speech, along with a local Python application that evaluates local commands. First, we had to sign to Google Cloud Platform and enable the API to be able to crate a new project for our AIY Voice Kit. Next step was to create credentials and download JSON [19] file with client secrets key. After finishing all these settings using the provided sample code, we could check if our speaker could recognise the voice.

Chapter 6

System Evaluation

6.1   Robustness

Unfortunately due to time constraints, we have not been able to test our application fully for robustness. For example, load testing and stress testing . However, what we do know from testing our application, as laid out in the following section, is that we have come across simple errors in GUI, slight glitches when it comes into text to speech conversion, and also in error margin when it comes to chatbot responses which is mainly caused by limited data set.

Another of the issues that has arisen while testing is that appliaction has to be forced to close by terminating the command prompt that has launched it.

6.2   Testing

Through continuous white box testing and regular black box testing we believe has not reach its full potential, mainly due to failure with Google AIY Kit, but also limitations when it comes to information about AI chatbots. However, it the project was satifactory when it comes to learning margin, overall outcome, and struggles we have overcame. To achieve these results we tested our chatbot application using white box testing in the following ways:

6.2.1   Google AIY Testing
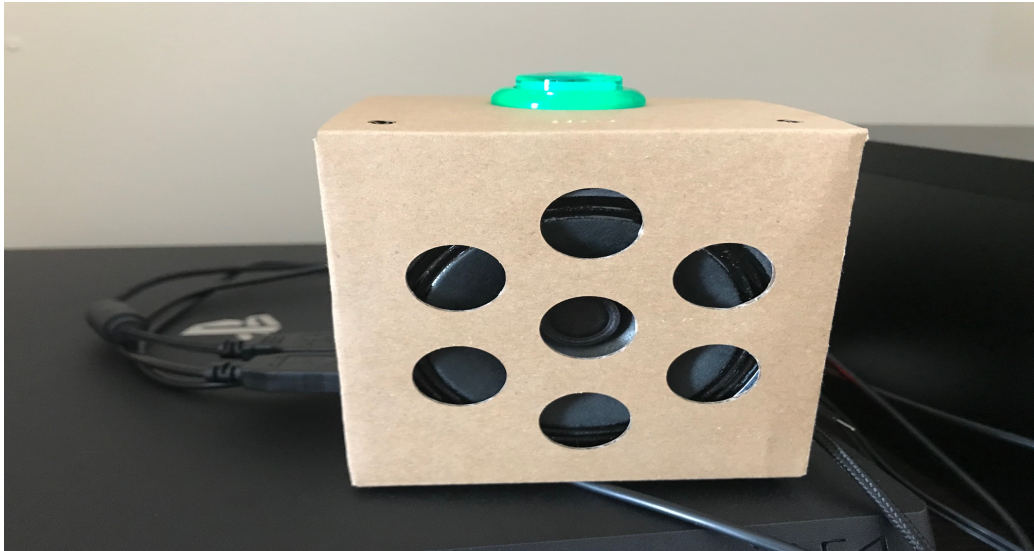
Figure 6.1: Google AIY Speaker

The first test we made on Voice Kit was to make sure that all hardware components are connected correctly. If the green LED light was flashing on the Raspberry Pi board that meant our device was correctly connected. The main function that we have tested has been the connection with Google Assistant and Cloud Speech-to-Text service. To test this feature, our sentence should start with "Ok, Google" and then question that we need to ask for example "Ok, Google what is 2 + 2?". If the speaker was able to reply, we had the confidence that the connection with Google Assistant API was correct.

6.2.2   Trial and Error

We have approached testing our application through trial and error method and divide and conquer. Where trial and error is is a fundamental method of problem solving. It is characterized by repeated, varied attempts which are continued until success, or until the agent stops trying. Then in computer science, divide and conquer is an algorithm design paradigm based on multi-branched recursion. A divide-and-conquer algorithm works by recursively breaking down a problem into two or more sub-problems of the same or related type, until these become simple enough to be solved directly.

We have divided our project into different parts, which are mentioned in Chapter 3.3. Those parts then were further divided into smaller sub objectives which were easier to manage. To make sure that our project would work as a whole we have tested each of the completed components through trial and error. That means that we have tested each component until we managed to get a satisfactory result.

6.2.3 Chatbot Testing

We have tested each of the chatbot components as mentioned in Chapter 6.2.2. The way that the testing was performed is that we have taken sample patterns from the intents.json and compared the result to expected result. We also have tested responses given shortened input, for example, if the pattern was "Which of the Dawn of the Dead was better?" we would shorten it to "Dawn of the Dead" and compare if results were familiar to those of the full sentence. Main reason for this testing was because we wanted to make sure that the neural network training went as planned.

6.2.4 Client-Server Connection Testing

Client-server was mainly about the connection. At the beginning there were slight issues with connection due to the socket TCP connection not being properly set up. After trial and error testing we have come to satisfactory results. Given that the connection could only be established if user allowed for it. Meaning one would have to enter hosts ip and port number. Next step of testing was to check if there would be no delay if number of users has increased. We have decided that for this prototype purposes five connections would be more than enough to keep the chatbot efficient.

6.3 Scalability

Our chatbot application was made to be scalable. Currently the data set that is supplied is limited in terms of content as it only covers certain topics. Expanding given data set should not be an issue, given that it's in reasonable size. If the data set will become too large, training it will take longer. To ease training with larger data set it is recommended to add more processing power.

Another aspect to our project which is worth a glimpse is server-client which only limits up to five connections to make sure that there's no slow traffic between user and chatbot exchange.

6.4 Results vs Objectives

When we have set out the requirements in Chapter 2.3 we were a bit too optimistic when it came to Google AIY Kit, as we have not met the objective to tailor it for our purposes. It was mainly due to poorly written documentation and dependencies that were out of date. These were also some issues with the licence where it stated that we could not alter the hardware and software. However, we did manage to overcome this difficulty by simply creating chatbot from scratch and adding all the components that we have learned during the process of this project.

With that said we can proudly say that rest of the objectives that were set out in Chapter 2.3 have been meet. We did manage to learn as much as possible about technologies that we have researched. We got to know how natural language processing comes into computer science and how it functions when processing large amounts of data. We have successfully produced a prototype chatbot which was trained on a given data set and supports server-client functionality.

6.5 Limitations

During the development process there were few limitation that we have came across that are worth mentioning, those are as follow:

6.5.1 Google AIY Kit Limitations

The main limitation is the unavailability of this product on the Irish or English market. If someone wants to start their adventure with AYI projects, they must order this set from the USA. Not every version of Voice Kit will provide power supply and microSD card. However, the most important limitation is that Google is misleading the users. Downloading the latest image for our Raspberry Pi we had to deal with the Speech-to-Text API [20] recognition and not as provided in the instructions with the Google Assistant API. Initially, it did not make much difference because the basic demo was working. However, when we wanted to modify the Voice Kit to our needs, then the problems started. Scripts that could be modified included methods that worked with older Google Assistant APIs. We took a lot of time to change the code in the script demo and did not bring any effects. Another limitation is that Google Cloud Speech-to-Text service is payable. Only 60 minutes a

month, it's free so not a good solution to use the Google Voice Kit as our everyday assistant.

6.5.2 Lack of Reliable Data Set

Soon after beginning testing phase of our chatbot we have found out that there was lack of reliable data sets. When it comes to traning the model and neural network data sets are really crucial as they determine what kind of chatbot we end up with. During the development we have also discovered that our chatbot could be used as a assistant. Given proper data set with appropriate patterns and responses chatbot could be pushed further where it could be used as automated customer service. That just shows how important data sets are when it comes to machine learning.
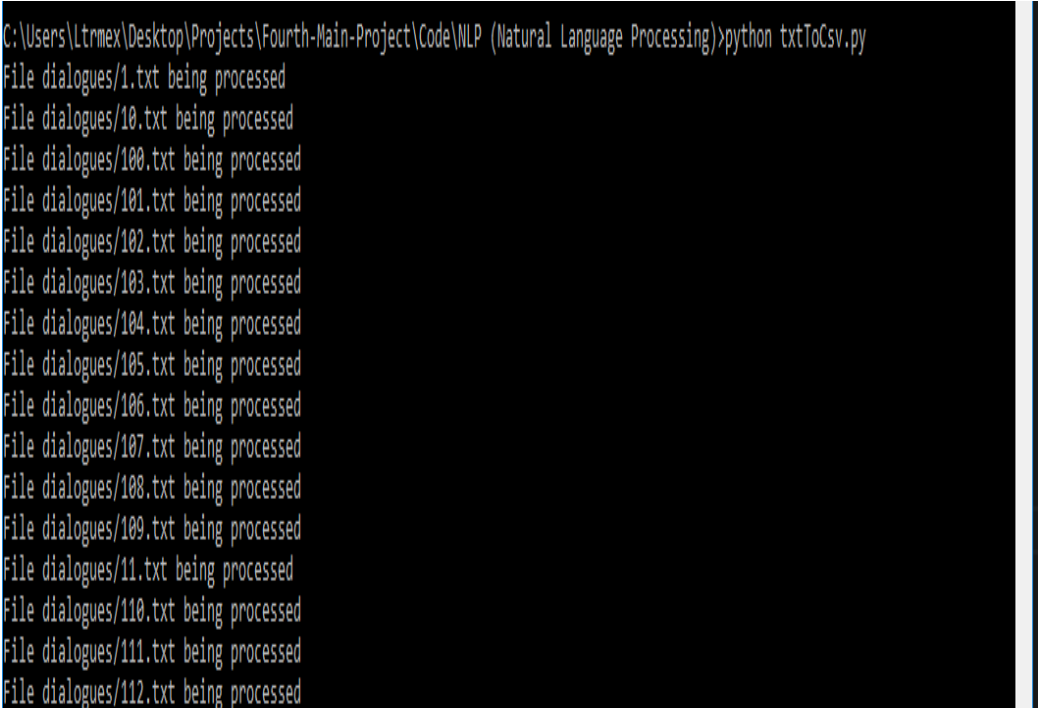
6.5.3 TkInter Graphical User Interface Limitations

Having tested using TkInter and GuiZero, and researched various other graphical user interfaces tool kits for Python programming language, we can conduct that most of them are limited. They only offer bare minimum when it comes to GUI development. Obviously we knew from the beginning that those tool kits weren't as efficient and reliable as GUI's when it comes to web development. However, we weren't aware that we were that limited when it comes to layout and design. From testing and research we can conduct that Python supplied tool kits for GUI development are mainly for instructions and control purposes, for example light switches or/and car control buttons.

6.6 Outputs


```
C:\Users\Ltrmex\Desktop\Projects\Fourth-Main-Project\Code\NLP (Natural Language Processing)>python get_data.py
04/25/2019 07:06:02 PM - Self dialogue corpus - Will write data to dialogues.
04/25/2019 07:06:02 PM - Self dialogue corpus - Will ignore assignments from these workers: ['A2GSIQYXC3SFMI', 'A34MRQ1OP87XC6',
'A3N8B6OSHHUUCI', 'AVFEF3TMBLERO']
04/25/2019 07:06:02 PM - Self dialogue corpus - Reading data into memory for topic action
04/25/2019 07:06:02 PM - Self dialogue corpus - Read 414 dialogues
04/25/2019 07:06:02 PM - Self dialogue corpus - Writing dialogues...
04/25/2019 07:06:07 PM - Self dialogue corpus - Done writing to dialogues/.
```

Figure 6.2: Data Extraction
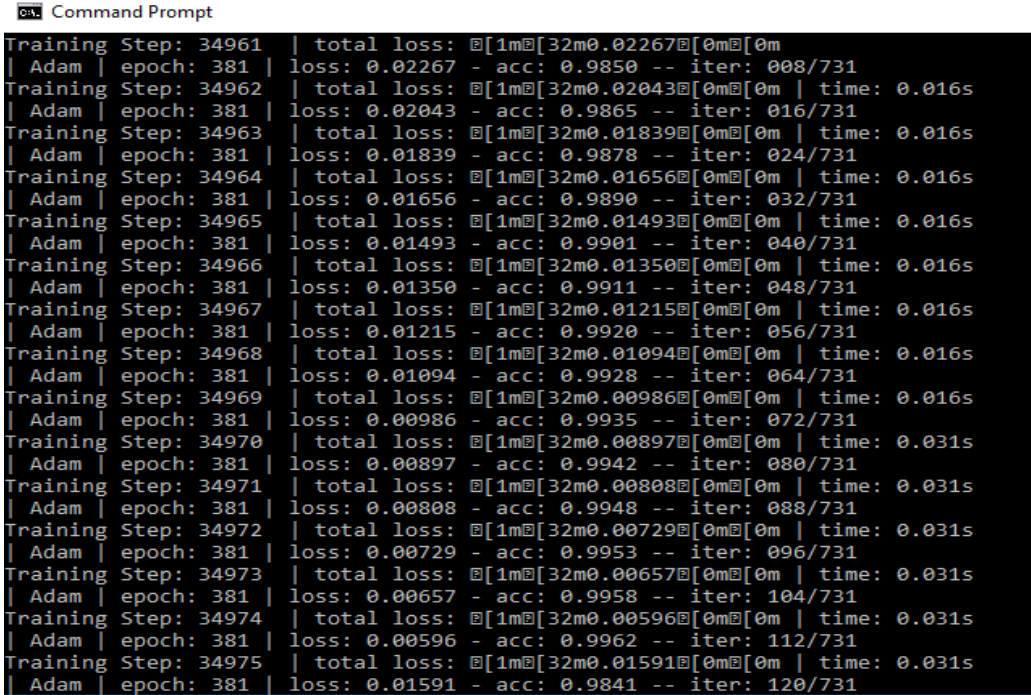
```
C:\Users\Ltrmex\Desktop\Projects\Fourth-Main-Project\Code\NLP (Natural Language Processing)>python txtToCsv.py
File dialogues/1.txt being processed
File dialogues/10.txt being processed
File dialogues/100.txt being processed
File dialogues/101.txt being processed
File dialogues/102.txt being processed
File dialogues/103.txt being processed
File dialogues/104.txt being processed
File dialogues/105.txt being processed
File dialogues/106.txt being processed
File dialogues/107.txt being processed
File dialogues/108.txt being processed
File dialogues/109.txt being processed
File dialogues/11.txt being processed
File dialogues/110.txt being processed
File dialogues/111.txt being processed
File dialogues/112.txt being processed
```

Figure 6.3: Data Conversion

```
C:\Users\Ltrmex\Desktop\Projects\Fourth-Main-Project\Code\NLP (Natural Language Processing)>python csvToJson.py
OrderedDict([('tag', 'Bourne'), ('patterns', 'I love the Jason Bourne franchise.'), ('responses', "Yeah, me too.-Matt Damon reall
y turned the spy genre upside down with that one.-How so?-He created a spy that had human flaws and wasn't perfect.-I can see how
 that is true.-The first one in the series is my favorite.-Mine too.-We really get to know Matt Damon's character for the first t
ime.-I agree and the action sequences were remarkable.")])
OrderedDict([('tag', 'Terminator'), ('patterns', "Can't believe Terminator 2 came out 25 years ago."), ('responses', "It's crazy,
 isn't it?-Still one of the better action movies I've ever seen.-The whole Terminator series was awesome, but T2 brought it to a
whole another level.-Arnold was a complete boss in that movie.-He was overwhelmed.-The T1000 was the superior machine.-Yeah, but
the T800 is no slouch.-Plus he looks like Arnold.-The rest of the series got pretty bad though.-Yeah, once James Cameron left the
 director's chair, it ran off the rails.-Think he'll ever get back in it?-I doubt it.-Doesn't change the fact they were great mov
ies.")])
OrderedDict([('tag', 'Expendables'), ('patterns', 'The Expendables was an excellent movie.'), ('responses', "Sounds familiar, who
's in it?-Sylvestor Stallone and Jason Statham.-Don't remember if I watched that one.-It's from 2010.-They were mercenaries.-Ther
es so many movies like that.-Yes, but not with those two in it.-True, then I guess I didn't see it.-It's an hour and fourty minut
es but it goes so fast.-Lots of action with those two.-Thats why.")])
OrderedDict([('tag', 'Face'), ('patterns', 'Did you ever see Face Off?'), ('responses', 'One of my favorites!-I know!-It was so g
ood!-I think what was most compelling is Cage and Travolta trading off.-Yeah, they both get to play the good and bad guy.-Very co
ol.-And the action was super intense.-I didn\'t care much for the "procedure".-Yeah, that was a bit gross.-I\'m shuddering just t
hinking about it!')])
```

Figure 6.4: Data Conversion

Figure 6.5: Model Training
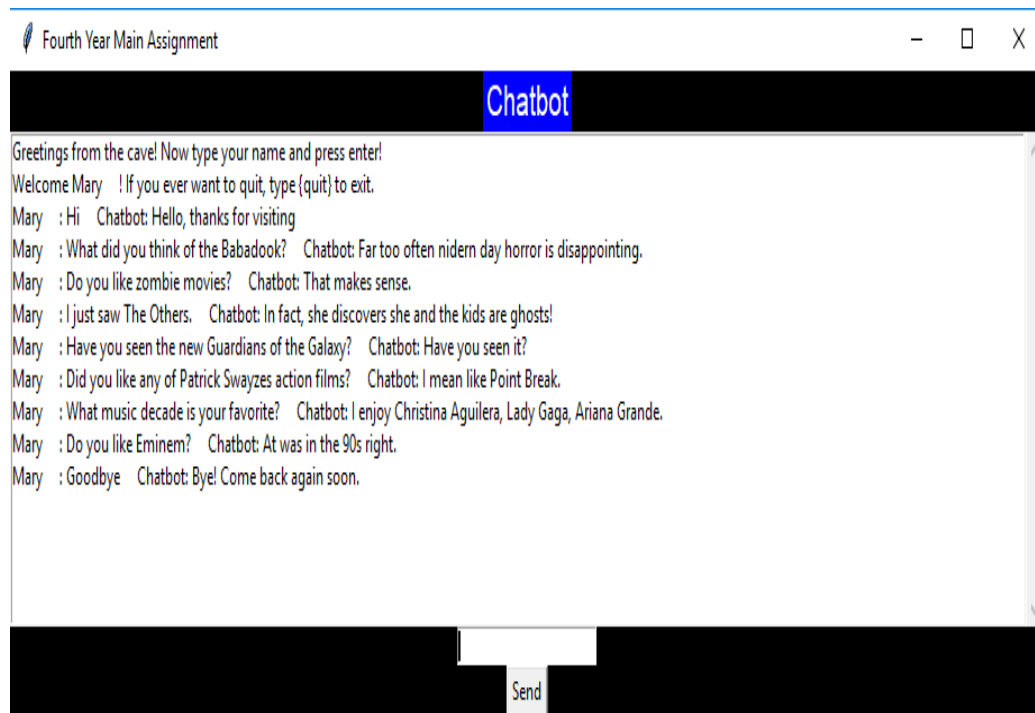

Figure 6.6: Client

Figure 6.7: Server

Figure 6.8: Chatbot

Chapter 7

Conclusion

Certainly, we can say that the project we chose was a challenge for us. Group work was a great experience, but besides our strengths, it revealed our weaknesses, but thanks to good communication and organization, we were able to complete this project according to plan. Choosing the agile approach, we had full control over each completed task and a plan for the next sprint, which facilitated our work in the group. During the four years of our education at GMIT, we developed assignment of various subjects, and we used the knowledge we previously acquired in them. In the case of this project, we put a lot of effort on research and learning about new technologies that we hope will be useful in our further career. Our main goal was to expand our skills in Python programming language. We wanted to deepen our knowledge regarding artificial intelligence and implement it in our project.

We chose to develop a chatbot for a reason. Artificial intelligence is constantly expanding and at the same time the demand for new technologies. The benefits of using chatbots on the enterprise side seem to be obvious. Thanks to chatbots, marketers reduce the costs of customer service, which allows them to intensify dialogue with consumers and, in many cases, improve its quality. Producers of goods and services have the opportunity to acquire new customers who spend most of their time using the Internet, and the chance to finalize the shopping route from advertisements displayed, e.g. in social media, increases. The development of technology that recognizes emotions will revolutionize dialogue with customers, make it become real personalized and much more engaging. Artificial intelligence will be able to understand a human will also increase the effectiveness of programmatic advertising, served not only according to the profile determined based on interests and behaviours in the network, but also the emotional state of Internet users. The development of chatbots will also accelerate other phe-

nomena of the digital revolution, such as spread of mobile internet and the related development of the Internet of Things. Devices that surround us will communicate with us through virtual advisers who will know more about us than our loved ones - family, friends and neighbours.

7.1   Summary of Context and Objectives

Our first idea was to use the Google Voice Kit and modify it to our needs. Due to the lack of documentation available and changes that Google introduced to the Cloud Speech-to-Text API, it was impossible. The mistake we made was the fascination of new technology which we did not investigate thoroughly. However, thanks to this experience, we came up with the idea of creating our own chatbot who will also use artificial intelligence.

The following goals were successfully achieved throughout the project:

- Research of the technologies such as: Artificial Intelligence, Neural Network, TensorFlow, Voice Recognition:

- Understanding what a Natural language processing is and how to use it.

- Improve Python programming language skills.

- Develop a graphical user interface for the chatbot.

- Make chatbot client-server based.

7.2   Future Development

For further development, we would like to implement a set of data that would allow the user to talk about various topics not necessarily related to the business for example, the weather. Another idea would be an algorithm thanks to which chatbot, after talking with the user, could define the range of interests, the average age group and potential interests. This project is suitable for a potential customer interested in selling services and products to clients from around the world.

7.3   Opportunities

During the development and testing process we have came to realization that more than a smart chatbot which could communicated and have casual con-

versations with the user, it would be more useful as a assistant. By assistant what we mean is that we could take our chatbot and tailor data set so that could be used as a automated customer service.

Given certain input patterns from the user it would response with a appropriate reply to the user query, thus making resolving issues, also known as troubleshooting, in more efficient way than just waiting in a queue for one of the employees to be available. The data the user is parsing could also be processed in certain ways, for example, lets say we have a customer that has an issue with an order during online shopping. We could resolve this issue by automated service, lets say that customer would connect to the assistant and describe its query, like "Order that I have received is not what I ordered", assistant then would response asking for user order id and date it was dispatched, then it could process that information and come up with a reasonable solution. This way online queries would be more efficient and connection to one of the employees would only be established once assistant wouldn't be able to solve customer query.

Bibliography

[1] TechRepublic. `https://www.techrepublic.com/article/61-of-businesses-have-already-implemented-ai/`.

[2] R. S. Wallace, "The anatomy of a.l.i.c.e.," *in Parsing the Turing Test (G. B. Robert Epstein, Gary Roberts, ed.)*, p. 81–210, 2009.

[3] "White box testing." `http://softwaretestingfundamentals.com/white-box-testing/`.

[4] "Black box testing." `http://softwaretestingfundamentals.com/black-box-testing/`.

[5] D. Ramel, "Popularity index: Python is 2018 'language of the year'." `https://adtmag.com/articles/2019/01/08/tiobe-jan-2019.aspx`, 01 September 2019.

[6] D. Crockford, "The json saga." `https://www.youtube.com/watch?v=-C-JoyNuQJs`, 28 August 2011. Retrieved 23 September 2016.

[7] "Google cloud products." `https://cloud.google.com/`, Retrieved 2017-06-02.

[8] Google, "Voice kit." `https://aiyprojects.withgoogle.com/voice/`.

[9] E. K. Steven Bird and E. Loper, "Natural language processing with python." `http://www.nltk.org/book/`.

[10] "Tensorflow." `https://www.tensorflow.org/`.

[11] "Sqlite." `https://sqlite.org/docs.html`.

[12] E. Rieuf, "History of mysql." `https://www.datasciencecentral.com/profiles/blogs/history-of-mysql`, December 16, 2016.

[13] IBM, "Ibm fortran program products for os and the cms component of vm/370 general information," *For users familiar with the predecessor FORTRAN IV G and H processors, these are the major new language capabilities*, p. 17, (first ed.), July 1972, Retrieved February 5, 2016.

[14] "Supercalc², spreadsheet package for ibm, cp/m," Retrieved December 11, 2017.

[15] "Data set." `https://en.wikipedia.org/wiki/Data_set`.

[16] "Definitions for data conversion." `https://www.definitions.net/definition/data+conversion`.

[17] "balenaetcher." `https://www.balena.io/etcher/`.

[18] "Google assistant sdk." `https://developers.google.com/assistant/sdk/`.

[19] "Json." `https://www.json.org/`.

[20] "Cloud speech-to-text." `https://cloud.google.com/speech-to-text/`.