***IT CODING FOR DATA SCIENCE 2020/2021***

***Project document***

***5014502 Savoldi Chiara***

***10/03/2021***

## **1.3 PLAgiarism (do not remove this section)**

##### **By uploading this report and the Project Python code, you declare that this is fruit of your work. It is acceptable to refer to Python functions/methods/tutorials found on the internet, but you must cite clearly the source of the material you have used (i.e. “I have used this tutorial** [**www.tutorial.com**](http://www.tutorial.com/) **to build my project”). You must be also able to show excellent understanding of what you have used.**

## **1.4 introduction**

*My project is related to the creation of a Chatbot, which is a software designed for interacting with humans in voice or text-based conversation. Since one of my passions, which I (and we) am (are) not able to “practise” in this period concerns travel, the discovery of places and the meeting of exceptional cultures, I decided to create a Bot, able to help tourists in a big city like Paris or London, competent to suggest the best places to eat, the sites of interest or some information about public transport. The bot can answer general questions starting from 'How are you', up to telling a joke. The name I chose, due to my lack of imagination, is “InfoTour”, where “info” stands for information, “tour” stands for tour around the city.*

*I think that a Chatbot is a simple example of artificial intelligence: its purpose is to show how a machine, after a great training, can be capable to provide quite comprehensive answers, like human, after a short question from the user. I know that people sometimes dislike human interactions: a Chatbot can be a good way to avoid them! I do not see myself in this thought because I am really extroverted.*

## **1.5 background theory and scientific foundations**

### **1.5.1 Background theory**

*My project required background on Python Classes and file management; basic knowledge of JavaScript, CSS, Flask, which is a framework for web applications in Python. Additionally, there is an extensive documentation regarding how a Chatbot works and how to train it, the associated functions, and libraries.*

*Flask has different decorators to handle http request: http protocol is the basis for data communication in the World Wide Web. There are different http methods: in particular, the most common method is “GET”, where a get message is send, and the server returns data.*

*I mainly used two websites and a Youtube video to design my Chatbot. I tried to figure out how to train the Bot to make its answers coherent to user's input questions.*

*No scientific basis was needed for my project. The main purpose of these software is to simulate human behavior: they are sometimes defined as “intelligent agents” and they are used for various purposes such as online help, to answer the FAQs of users who access a site; some software uses sophisticated language processing systems, others just scan the keywords in the input window and provide a response with the most matching with the keywords. The main qualities required are a good knowledge of the areas mentioned above and the willingness to train your “robot” in the best possible way.*

## **1.6 technical analysis**

### **1.6.1 Python libraries and specific functions used**

*Here is the list of libraries used:*

### *Import os*

### *From chatterbot import ChatBot*

### *From flask import Flask, render\_template, request*

* *From chatterbot.trainers import ListTrainer, ChatterBotCorpusTrainer*
* *From nltk.chat.util import Chat, reflections*

*Python Os module allows us to interact with the operating system in several ways. In particular, I used the method os.getcwd() which returns the current working directory of a process. The os.path.join() method in Python join one or more path components: it concatenates various path components with one directory separator (‘/’) following each part, except the last path component. The last method is os.path.isfile(), which is able to check whether the specified path is an existing regular file or not.*

*ChatterBot is a library that easy provide automated responses to a user’s input. ChatterBot can use a machine learning algorithm to produce different types of responses. I create an instance of the class Chatbot for defining my Bot.*

*Flask is a framework for creating web applications, which includes a set of programs with the aim of facilitating the creation of web services such as web servers, APIs and more.*

*I defined an instance of the Flask class to create a new app, defining a route to the home address ('/'). My goal was to define a function that would display an HTML page in the browser. To do this, I used the render\_template function, which takes the name of the HTML page as a parameter and automatically sends it to the browser. From our browser by navigating to the address localhost: 5000 we will see my html page.*

*The flask Request object contains the attributes of the URL request. The args attribute is a dictionary holding arguments from the URL. The get() method will “get” an item from a dictionary or return an error if key is not found.*

*ChatterBot Corpus Trainer allows the ChatBot to be trained using data from the ChatterBot dialog corpus, to simply make the communication between the Bot and the users. Training with general corpus will give more conversation to the Chatbot.*

*List trainer allows the ChatBot to be trained using a list of strings, which represents a possible conversation between the Bot and every user.*

*Chat is a class that contains the logic used by a Chatbot. Reflections is a dictionary that includes a set of input values and its corresponding output values. It is an optional dictionary that we can use. I have also created my personal dictionary “responses”.*

### **1.6.2 Python code**

## *I found two interesting libraries to implement a Chatbot, for this reason I tried to create it in both ways.*

## *CHATTERBOT:*

## *I divided my project into four Python scripts:*

## *-* ***Initialize\_interface.py****: first, I defined the function produce\_chatbot\_hp, which receives in input the name of a file. Its purpose is to create an HTML file that represents my Bot's introduction webpage. My idea is to provide a short presentation of the Bot, offer a space to implement a short chat with “him” and introduce a sort of choice: if you are in Paris, admire the image and click on the link for information on the history of Paris and its beauties, if you are in London vice versa. The purpose was to create a 'user-friendly' page, pleasing to use’sr eyes. The second function create\_chatbot\_interface check whether at the specified path is an existing regular file or not: if yes, it prints the message 'The file already exists', otherwise it creates the HTML file.*

## *-* ***Initialize\_chatbot.py****: I created the start\_conversation function which requires (chatbot) as parameter. I settled a default message from my Chatbot, to be printed every time the function is called. First, it creates and opens the 'SavedConversation' file, which will save every conversation between users and Chatbot.*

## *The get\_response(input\_item, conversation\_id=None) returns the bot's response based on the input. If the user does not write 'Goodbye' he can talking to the bot. The second function, train\_chatbot (chatbot), trains the Chatbot using a sort of list of lists variable. I create short conversations “one query, one answer” and add them to the list. In this way, is going to be easier to train the Bot. I also wrote some lines of code to train the Bot with general corpus. It will give more conversation to “him”.*

## *My chatbot is defined as an instance of the class ChatBot(). The only parameter required is the name of the Bot, but I can also specify some parameters, that represents the level of personal customization: for example, #output\_adapter = "chatterbot.output.OutputAdapter", which allows the Chatbot to communicate through the terminal and allows a user to type into their terminal to communicate with the Chatbot.*

## *-* ***Conversation.py*** *which includes a list of lists, each containing the possible user input and the response that the Bot should give.*

## *-* ***App.py****: I defined the application to allow a conversation with the Bot from my web page. To the initial presentation web page, I added some lines of Javascript and Css code. I created my Bot, the render\_template function, which takes the name of my HTML page as a parameter and automatically sends it to the browser. Finally, the get\_bot\_response() function which, if the request is successful, applies the usual get\_response() function of the Chatterbot library to the Chatbot, to receive a response after a user input from the web page.*

*-* ***CSS and JS****: I first added a background image to my webpage and defined its size which must cover all the web page. For example, I have used the class attribute to point to a class name in my css file. I also used the class attribute in my JavaScript to access and manipulate elements with the specific class name. For example, we have two <div> elements with a class attribute with the value of "chatbox". All of the two <div> elements will be styled equally according to the .chatbot style definition in the css file. For the JS part I defined the function getBotResponse, which does not receive any parameter and, first, it returned the value attribute of the element with ID = #textInput. I showed the previous sent message as a message in the interface and cleaned the text box of the input message, so I clean that box to not make the person delete the text to send another message. The link that calls the file.css is <link rel = "stylesheet" type = "text / css" href = "file.css">, while the JS code is insert directly in the "body" section of the html code of the page, enclosed in these tags: <script> code </script>*

## *NLTK:*

## *The work on this library was very superficial: the Chatbot was defined as an instance of the Chat class to which I passed the default 'Reflections' dictionary and my personal 'responses' dictionary. If the user does not type 'quit', with the converse() function it is possible to talk to the Bot. I have set multiple answers for the same input phrase. Furthermore, if I write "My name is Chiara", the Bot will take the word "Chiara" and reuse it in its answer within the phrase "Hi Chiara, how are you?", to make the Bot less monotonous and able to respond differently to each user. The Bot will match the string* ***r"possible user input"****:*

## 

## *The asterisk inside the round brackets represents a reggae or regular expression. It means "you can put anything here". The string in brackets with all pipes indicates that you can get an answer with both the possible options indicated.*

## **1.7 conclusions**

### *The result is not exactly what I expected because is complex creating a real tourist guide; it would be interesting to implement an InfoTour that is able to intercept the real position of a person in any European city. The script was not very complex at the instruction level because the convenience and beauty of Python are the libraries that it offers: for example, the get\_response() function, applied to my bot was already implemented and I simply have used it. Another aspect against my work is the fact that, by creating a list of possible questions and answers between the user and the Chatbot, I will not be able to cover the vastness of questions that can be made to the Bot and the vastness of possible inputs from the users. For this reason, a good Bot is one that provides an exhaustive answer to a user's questions. I think this is the limit of these software: it will never be possible to include a Bot that, for the 100% is able to work with all the possible questions posed by a human.*

### *I think that I might go more in-depth with a thesis work on this topic, to analyse well the existing libraries for the creation of Bots, and to improve knowledge at Python level: through the internet I discovered more intelligent management, for example to read or write a file and some tricks to make the code as general as possible.*

### **1.8 BIBLIOGRAPHY**

[*https://chatterbot.readthedocs.io/en/stable/*](https://chatterbot.readthedocs.io/en/stable/)

[*Python os Library Functions - JournalDev*](https://www.journaldev.com/30205/python-os-library-functions)

*https://www.youtube.com/watch?v=tSjR7bk1Y9U*

[*https://flask.palletsprojects.com/en/1.1.x/api/*](https://flask.palletsprojects.com/en/1.1.x/api/)

[*https://developers.google.com/web/updates/2013/01/Voice-Driven-Web-Apps-Introduction-to-the-Web-Speech-API*](https://developers.google.com/web/updates/2013/01/Voice-Driven-Web-Apps-Introduction-to-the-Web-Speech-API)

[*https://www.w3schools.com/jquery/*](https://www.w3schools.com/jquery/)

*https://www.edureka.co/blog/how-to-make-a-chatbot-in-python/*