**Introduction to AI COM526 Software Prototype and report (AE2)**

|  |  |
| --- | --- |
| Name: | Ismail Kefokeris  Q15328350 |
| Choice of prototype: | Chatbot |
| Team member name: | AGBOLADE DAVID |

**Contents**

[**Introduction** 3](#_Toc91994963)

[**Need for the chatbot** 3](#_Toc91994964)

[**Statement of the problem** 3](#_Toc91994965)

[**Aims and Objectives** 3](#_Toc91994966)

[**Objectives:** 3](#_Toc91994967)

[**Literature Review** 4](#_Toc91994968)

[**Proposed Solution** 4](#_Toc91994969)

[**Prototype Design** 4](#_Toc91994970)

[**Prototype Development and AI Algorithms used** 5](#_Toc91994971)

[**Libraries:** 5](#_Toc91994972)

[**Code Walkthrough** 6](#_Toc91994973)

[**Train.py** 6](#_Toc91994974)

[**Chatbot.py** 8](#_Toc91994975)

[**How to run** 10](#_Toc91994976)

[**Evaluation** 10](#_Toc91994977)

[**Limitation** 10](#_Toc91994978)

[**Conclusion** 10](#_Toc91994979)

[**Bibliography** 11](#_Toc91994980)

# **Introduction**

I have been tasked with building a working prototype chatbot together with my peer David. When developing this prototype, I must be able to consider issues that may be relevant to the chatbot. Due to covid more people are turning to online. Businesses have seen the importance of making themselves accessible through the internet, instead of hiring new people to take care of things such as appointments its more cost effective to hire a small team to manage and build a bot which can run all day every day (Takeshi, Tatsuya, Atsuko, Tsunetsugu, & Tomoo, 2019) For an interface to be noted as acceptable according to “The Trouble with Chatbots: social skills in a social world” it must display social intelligence and must participate in the social hierarchy. (Wallis & Norling, 2005) Showing some of the issues I will face in the design and building of the conversational agent. Apart from social skills chatbots face the obvious concern, which is privacy, what happens with the data? is it collected? And what systems are in place that my data is safe.

## **Need for the chatbot**

More and more businesses are going online and require 24/7 customer service to keep up with demand and to enable them to go global. The chatbot is one of the best ways to allow businesses to answer simpler more common questions from customers fast without having to pay real people to sit behind a computer waiting for oncoming messages.

## **Statement of the problem**

The chatbots purpose is to reduce time spent on booking appointments either through visiting or calling an establishment. The chatbot will allow for automatic appointment booking or reschedule and assist both user and admin.

## **Aims and Objectives**

The aim for this project was to create a simple yet somewhat intuitive AI chatbot to help businesses book appointments bypassing the need for customers to call in and speak to someone to book an appointment.

### **Objectives:**

Able to function as a help service for booking appointments.

Rule based chatbot with scripted actions based on keywords (intents).

Able to respond with a valid response.

# **Literature Review**

AI chatbots are software that can have conversations with humans, they enable a user to interact with anything from a device to a service such as booking an appointment. AI chatbots accomplish this using NLP (natural language processing). Chatbots are becoming more and more popular for many reasons such as working 24/7. These conversational agents have become almost ubiquitous without society, for example we have personal assistants on mobile devices and even bots that answer the phone to help direct you to the right agent or bots to give advice whether its buying something, if you require legal advice or even therapy in some cases. (Iulian, et al., 2017) (Rich, 2018).

There are a variety of models for chatbots, a model very similar to the one that is used in my prototype is the knowledge base-based question answering model which according to (Iulian, et al., 2017) is used in Evi an amazon question answering web-service and is designed to answer factual questions.

Over time humans will adapt and become more dependent on human/machine partnerships. We will eventually reach the stage where voice may replace typing and add new things such as gestures, eye-tracking or other types of interfaces. With machine learning AI will be inserted into not only businesses with functions such as task management or research but also into non-work-related activities like shopping advice, banking, cooking. A lot of which we already see now. (Daniel & Olivier, 2019)

# **Proposed Solution**

The chatbot needs to use machine learning to provide 24/7 service for customers to book appointments, it must include basic NLP to answer simple questions from the user like Evi from amazon (Iulian, et al., 2017) but also potentially understand tone, mood, etc. To ensure the chatbot returns adequate results we would require a large data set to train it. We would not store any user data to avoid both ethical and legal issues but instead train the chatbot on data collected and provided by myself or others who have provided consent. To further the bot, we could potentially collect questions asked by users anonymously to see what is frequently asked and if we can provide an answer for it.

# **Prototype Design**

The prototype will be run from two python files making use of a variety of libraries. First file would oversee training the chatbot, this would be run before. Train.py file will take a dataset (intents file) and run the program creating a model, this model uses neural network with a single hidden layer. The second file will be the interactive chatbot file which will run the script allowing the user to interact with the chatbot model.

# **Prototype Development and AI Algorithms used**

Both Jupyter notebook and PyCharm were used, in addition to this a variety of libraries were also utilised:

### **Libraries:**

* TensorFlow
* Random
* Pickle
* NumPy
* Json
* NLTK

Building the chatbot prototype started with researching other chatbots that were made public, this included tutorials or GitHub repos or completed code. We potentially could have utilised chatterbot library which may have made the programming easier, but we decided we wanted something more advanced and wanted to attempt using neural networks. Other ideas included implementing facial recognition with OpenCV and/or voice input/output.

Both me and David worked on a chatbot individually, we have ultimately decided to work on David’s chatbot as he completed it before I did and its potentially simpler. My chatbot would have used the machine learning library based on the Torch library called PyTorch. PyTorch is very powerful and helped me build a sophisticated chatbot, as improvement to the current bot I would consider it again.

The three files that are used in the prototype:

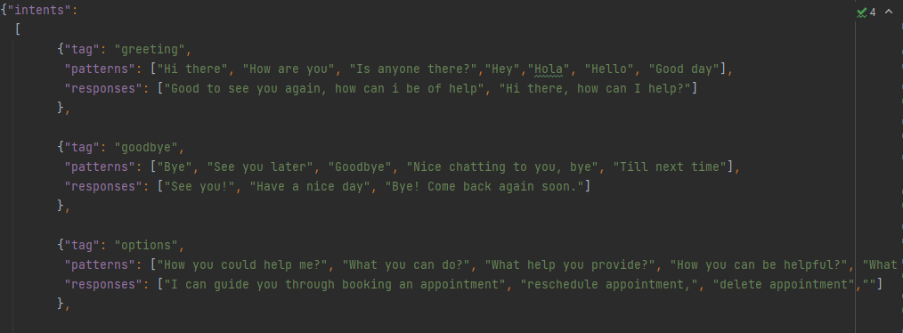
Intents.json – Contains text data.

Train.py – Python file contains the script required to read the intents json file and make the model.

Chatbot.py – Python file contains the script required to read the model. Also allows the user to interact with the chatbot.

### **Code Walkthrough**

Firstly we created a file which would hold the data that would be used to train the chatbot (intents.json).

Inside intents.json is a list of dictionaries which I will be calling different categories, each category will have three pieces of information “tag”, “patterns”, and “responses”.

#### **Train.py**

1. Import all the required modules, we are using nltk to process the sentences and create bag of words, tensorflow for the machine learning aspect and to work with neural networks.

Graphical user interface, application, website

Description automatically generated

We use a lemmatizer from NLTK the “WordNetLemmatizer” function and load in intents.json. Finally the script will define variables for later use, Words = all the words from the intents, Classes = all the tags, docs = combinations, and ignore = characters that shouldn’t be considered

Text

Description automatically generated

Now we will iterate over intents, and inside that we will find each pattern and then tokenize it. By tokenizing we split up the sentence into individual words. The script will then include what has just been tokenized into the pre-defined words list and the docs list will take the tokenized sentence in the form of a tuple together with “tag”, so we know where the word list belongs.

We now check if the tag is not in the classes list, we created earlier, if so then we simply add it.

Text

Description automatically generatedNeural Networks require numerical values, so we need to represent our words as numerical values. We do this by first creating an empty list which is where we will be appending all the numerical values, we loop through the doc list which contain both the tokenized sentence and the tag, we extract just the sentence using “doc[0]” and put it through lemmatize, we then look at each word in the list words and append 1 every time the word is inside our lemmatized word\_patterns list otherwise a 0. Finally, we get the class index and find which position it is in to set to 1 then append both the numerical values of the sentence and the tag into train.

Text

Description automatically generated

Now the script will shuffle the train data and turn it into an np array as that’s what tensor flow likes to use. We also set the train data to x list and y list which are our sentences and tags.

Text

Description automatically generated

The next step is to build the neural network model. The Dense layers are three hidden layers the first is an input layer the second is a hidden layer and the third(output) is the number of classes with SoftMax applied to allow us to get probabilities of each class.

Text

Description automatically generated

It will set a few other settings and compile the model before setting the number of epochs (how many times we will feed the data into the neural network), batch\_size (how many at once) once it’s done the model will be saved as an h5 file and end.

#### **Chatbot.py**

Text

Description automatically generated

Chatbot loads all the previously saved pieces of data for use in the script.

Text

Description automatically generated

We define a function called clean\_sentence which will clean up the sentence using tokenize and lemmatize as we did in the train.py

Text

Description automatically generated

We define a function called words\_box which will convert a sentence into what is known as a bag of words, a list of 0s and 1s which will indicate if the word is there, this is very similar to what was done in train.py

Text

Description automatically generated

We define a function called prediction which will utilize both functions previously defined and will return the tag name of the sentence the user has input as well as the probability of it being that tag.

Text

Description automatically generated

We define a function called get\_reply which returns out a reply based on the tag that it has been provided from the previous function and picks a random response in the responses in that specific tag.

Text

Description automatically generated

Finally, this is the main chatbot loop which will allow the user to interact together with the trained model.

### **How to run**

First: “ **pip install -r requirements.txt** “

Second: **Run train.py**

Third: **Run Chatbot and Enjoy**

# **Evaluation**

Throughout the code to understand what was happening when it was happening I wrote up a few print statements to help me, for instance when I was in the process of creating the bag of words and didn’t fully understand it I took a deeper look with a few print statements to understand what the different parts were.

I also tested with the number of epochs and the batch size to see how much better the chatbot worked, I first started with 200 and slowly increased, I found that 15000 may be a little over kill but it’s the number that worked best for me, although the chatbot still makes a few mistakes.

To ensure I was getting the right response I would give the chatbot vague sentences and see if it could understand what I meant, unfortunately I think it needs more training.

# **Limitation**

The level of understanding the chatbot has is very limited to what we have provided in the intents file. Additionally, you can’t really have a conversation with the bot as its more of a question-and-answer type of bot rather than a conversation type.

# **Conclusion**

In conclusion the prototype has been successful in what we intended it to be, but for the future I would improve it by adding more sophisticated features such as facial recognition, text to speech, and be able to understand speech. I would also find a better way of supplying training data to the bot as manually writing up an intents file takes an enormous amount of time and doesn’t get great results.

# **Bibliography**

Daniel, N., & Olivier, B. (2019). *Human/Machine : The Future of Our Partnership with Machines.* Kogan Page, Limited.

Iulian, S., Chinnadhurai, S., Mathieu, G., Saizheng, Z., Zhouhan, L., Sandeep, S., . . . Vi. (2017). A Deep Reinforcement Learning Chatbot. *ArXiv*, 40.

Rich, M. (2018, March 12). *What are chatbots? And how is AI making them better?* Retrieved from IBM: https://www.ibm.com/blogs/industries/what-are-chatbots/

Takeshi, K., Tatsuya, I., Atsuko, M., Tsunetsugu, M., & Tomoo, I. (2019). A Chatbot System for Mental Healthcare Based on SAT Counseling Method. *Mobile Information Systems*, 11.

Wallis, P., & Norling, E. (2005). The Trouble with Chatbots: social skills in a social world. *Virtual Social Agents*, 29-36.