

System Description

My chatbot's primary function is to recommend books to a user. It uses two knowledge bases to do this. One for fiction books and the other for non-fiction books. These knowledge bases contain a list of books categorized by genres. An example is shown later in the knowledge base appendix section of this report. The chatbot is created by using a neural network to accurately predict suitable responses based on the user's input. This was done by training the neural network on a json file of intents that can be viewed in the project directly. The chatbot is also able to remember past users by looking up their name in a user_database that is stored as a pickle file. When the program runs this pickle file is stored as a dictionary for look up optimization.

That chatbot was created with the use of several NLP techniques that we have discussed throughout the semester. I used the nltk lemmatizer to lemmatize words. This was done to increase the training performance of the neural network. I also used nltk sentence and word tokenizer to tokenize sentences/words from the knowledge bases as well as for the user's input. I also used the concept of pre-processing sentence data to improve performance of the nltk functions. For example, removing punctuation from the sentence and lowercases the entire sentence. Lastly, I pulled from the latest portion of the class by using a neural network in conjunction with these natural language processing techniques. To do this I used keras.

Diagram Logic (List Format)

- Load knowledge bases pickles
- Store pickles in dictionaries
- Open intents.json file
- Lemmatize the word in intents.json
- Pickle the list of words and classes (originally did this because I planned to create a separate main.py file but ended up leaving everything in one file)
- Create training data
- Build neural network model
- Open user database (pickle file/dictionary)
- Take in user input
- Predict the classes and calculate probabilities
- Generate the bot's answer based on the predicted class
- Bot responds to user
- Store user's response as it comes. For example, right after user gives the bot their name, store user input as "user_name), etc.

- If the user's name already exists in the database then prompt the user that they already exist. Display the user's past preferences/list of recommendations
- If the predicted class is "recommend", prompt for fiction or nonfiction preference for the user and store it in the user database.
- Randomly selected a book from the knowledge base
- Store random selection in the user database under "list of recommendations". This is done to prevent repeated recommendations
- Repeat until the predicted class is "bye". If the predicted class is "bye" then the chatbot will choose a goodbye prompt to display to the user and store the user's information.

Sample Dialog

First Time User

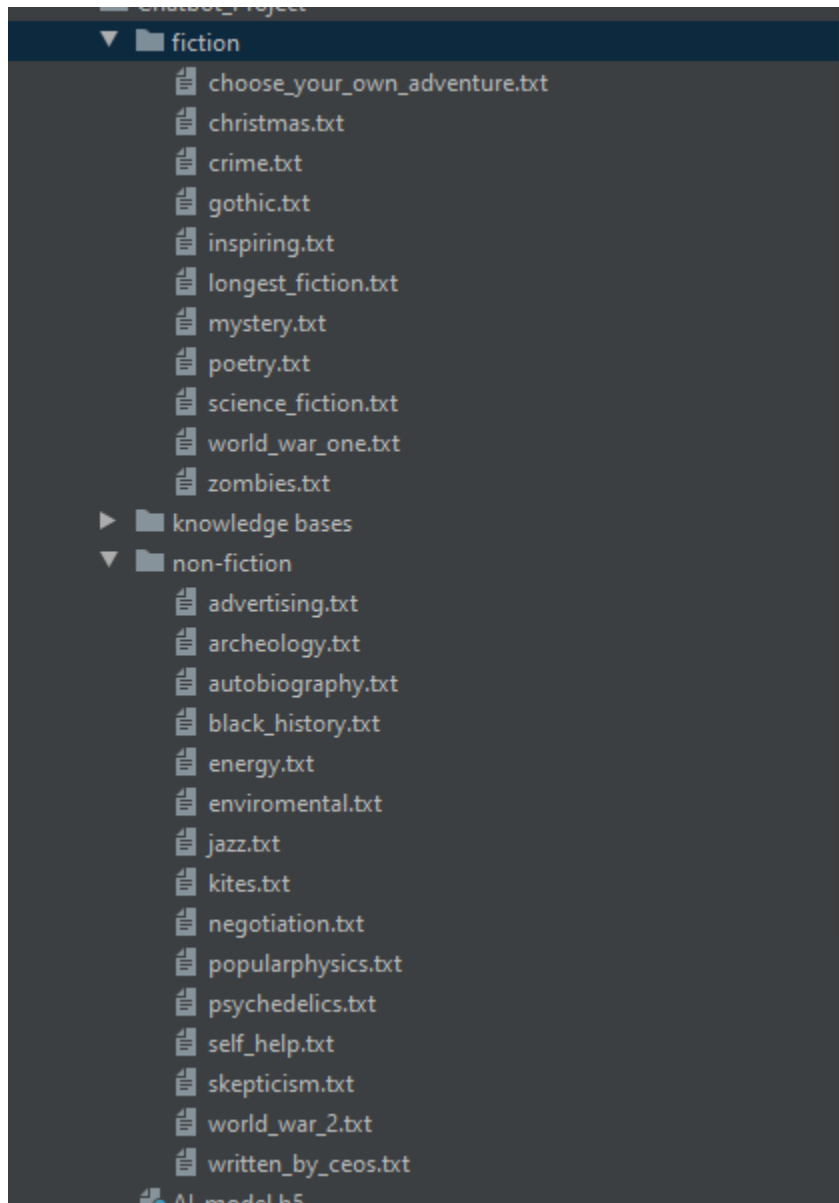
```
Model Successfully Created.  
BOT has entered the chat  
  
BOT: Hello, I am a bot that can recommend books based on your preferences. What is your name? anthony  
BOT: Hello, anthony  
  
hi  
Hello, what can I do for you?  
what can you do?  
I am a bot I can recommend books for you, just ask me to  
okay, recommend a book please  
Do you prefer fiction or non-fiction? fiction  
Here are fiction genres ['choose your own adventure', 'christmas books', 'gothic fiction', 'inspiring novels', 'longest fiction novels', 'mystery novels',  
What genre interest you? (Type as shown) inspiring novels  
Here is my book recommendation from that genre: Orlando by Virginia Woolf  
bye  
See you later, thanks for visiting. User data has been saved  
  
Process finished with exit code 0  
|
```

Returning User:

```
3/22 [=====] - EM: 0.3 - loss: 0.1977 - accuracy: 1.0000  
22/22 [=====] - 0s 225us/step - loss: 0.3268 - accuracy: 0.8636  
Model Successfully Created!  
BOT has entered the chat  
  
BOT: Hello, I am a bot that can recommend books based on your preferences. What is your name? Anthony  
Oh, Hello again anthony  
Here is your information from last time  
{'books': ['Orlando by Virginia Woolf'], 'genre': ['inspiring novels'], 'preference': 'fiction'}  
bye  
See you later, thanks for visiting. User data has been saved  
  
Process finished with exit code 0
```

Knowledge Base Appendix

The knowledge base for the chat bot is broken down into two separate entities, fiction and nonfiction. There is a directory in the project for each. As you can see from the screenshot, each directory has files for the respective genres. For example, the fiction directory contains the genres, science fiction, poetry, etc.



These files simply contain a list of books for the respective genre, as well as the author of the book. For example, the first three lines of the crime.txt file read:

The Daughter of Time by Josephine Tey
The Big Sleep by Raymond Chandler

The Spy Who Came In From the Cold by John le Carré

Next, I created two dictionaries to store these lists of books. How I programmed this can be seen under the “knowledge bases” directory in the files “get_fiction_knowledge_base.py” and “get_nonfiction_knowledge_base.py”. Essentially, each dictionary looks like this. {genre1: [list of books], genre2: [list of books]}. These dictionaries are pickled in the “get_fiction_knowledge_base.py” and “get_nonfiction_knowledge_base.py” and read in from the main function. The reason for the separation was to improve runtime. Here is a snippet of the output for the fiction dictionary.

```
kites.txt 146
negotiation.txt 146
popularphysics.txt
get_fiction_knowledge_base() x
C:\Users\antma\anaconda3\python.exe "C:/Users/antma/NLP_Code/Homeworks/Chatbot_Project/knowledge bases/get_fiction_knowledge_base.py"
{'choose your own adventure': ['The Cave of Time by Edward Packard', 'Journey Under the Sea by R. A. Montgomery', 'By Balloon to the Sahara (reissued as Danger in the Desert) by Douglas Terman', 'S
Process finished with exit code 0
```

Sample User Models

The user models are stored as dictionaries of dictionaries where the key is the user's name and the value is a dictionary of the user's information. Here the general format:

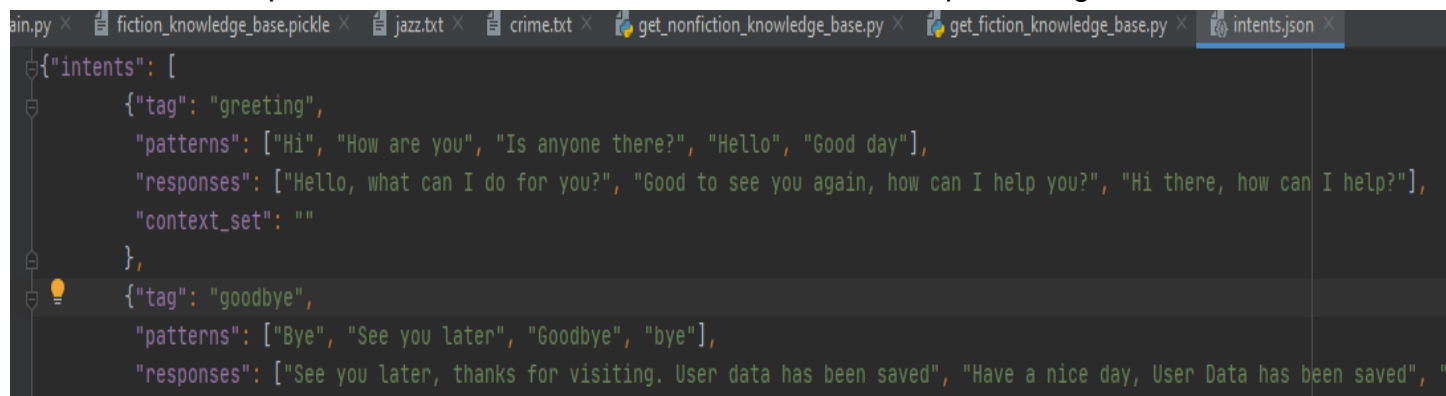
name: {"prefer fiction or nonfiction": fiction, genre: [list of preferred genres],
"recommended books" : [list of books previously recommended by the bot]}

Here is an example for the user model for “Anthony”:

```
Oh, Hello again anthony
Here is your information from last time
{'books': ['Orlando by Virginia Woolf'], 'genre': ['inspiring novels'], 'preference': 'fiction'}
bye
See you later, thanks for visiting. User data has been saved
```

Evaluation of The Chatbot | Analysis Strengths and Weaknesses

Since the chatbot was created using a neural network by training it on a file of intents (intents.json) the bot is pretty flexible. The user does not need to type input in the console in a very specific format for the bot to understand and generate a response. For example, we can look at the greetings intents. The neural network is trained to recognize the patterns “Hi”, “How are you”, “Hello”, etc. The network will correctly estimate the probability that the user’s input (Hello there bot) falls under the “greeting” intention and thus, generate a response that makes sense, “Hello, what can I do for you?”. Another strength of this bot is that it is capable of keeping track of past users. This means that it can avoid making repeated recommendations. The bot can also keep track of the user’s preferences such as “fiction or non-fiction” and “preferred genres”.



```
{
  "intents": [
    {
      "tag": "greeting",
      "patterns": ["Hi", "How are you", "Is anyone there?", "Hello", "Good day"],
      "responses": ["Hello, what can I do for you?", "Good to see you again, how can I help you?", "Hi there, how can I help?"],
      "context_set": ""
    },
    {
      "tag": "goodbye",
      "patterns": ["Bye", "See you later", "Goodbye", "bye"],
      "responses": ["See you later, thanks for visiting. User data has been saved", "Have a nice day, User Data has been saved", ""]
    }
  ]
}
```

As far as weaknesses, the bot struggles with the “thanks” intents. This can be seen when the user says “thank you” after the bot makes a recommendation. For some reason the bot will respond with a greeting tag such as “Hi there, how can I help?”. Another weakness is that the bot does not have a great user interface. It is simply run in the console. To improve this aspect a GUI could be created for the bot. Or the bot could be hosted on a service like Discord. Lastly, the bot does not save the user’s information into the user database until the user says “bye”. It would be better if the bot stored this information as it received it from the user to prevent “loss of progress”.