Jeopardy Chatbot

(The revenge of Alex Trebek)

Goals/Reasoning

As many people get older, loneliness and depression occur, in part due to the lack of stimulating conversation. Natural Language Processing (NLP) proposed one solution which was the historical ELIZA chatbot created by Joseph Weizenbuam at MIT. The ELIZA, although not intended to do so, created a confessor and therapist by provoking conversation. In the same spirit, the purpose of this Chatbot was to create a means by which people can actively use their minds and entertain themselves.

In many senior assisting, physical therapy, and mental health centers will televise stimulating entertainment. The Bot is based on the popular television game show series, Jeopardy. The bot's name is Alex, in honor of the longtime Jeopardy host, Alex Trebek. The original intent was to have the Alex-bot, to have stimulating conversation as well as to provide a means of playing trivia-based game.

Also, the original intent of the Chatbot was to be an online webapp on the Python Flask platform. However, due to time constraints and personal problems, the full web app was not completed. However, the skeleton is still available for future completion.

System Description

NLP techniques.

Corpus and Knowledge Base formation was the first step in the creation the Jeopardy Chatbot. Although, a WebCrawler was previously created, the knowledge based obtained surrounded around ChatGPT. Although quite interesting, this did not seem to fill an entertaining and informative aspect that Chatbots require. After some review, I was able to find an interesting knowledge base of over 250,00 Jeopardy questions, from reddit, in the form of JSON file. (https://www.reddit.com/r/datasets/comments/1uyd0t/200000_jeopardy_questions_in_a_json_file/) The knowledge base includes the questions, point values, categories, date the question was aired, show number, and of course the correct response.

I was concerned about using this as the only knowledge base for the chatbot as I did not just want to make assumption on questions presented by the Jeopardy game. As such, I was able to find several knowledge bases on response to chatbot and well as intents in JSON format style on

Kaggle. With sincere apology, I made the mistake to confirm which entry I pulled from and when I went back in order to cite the depositor, I could not find it.

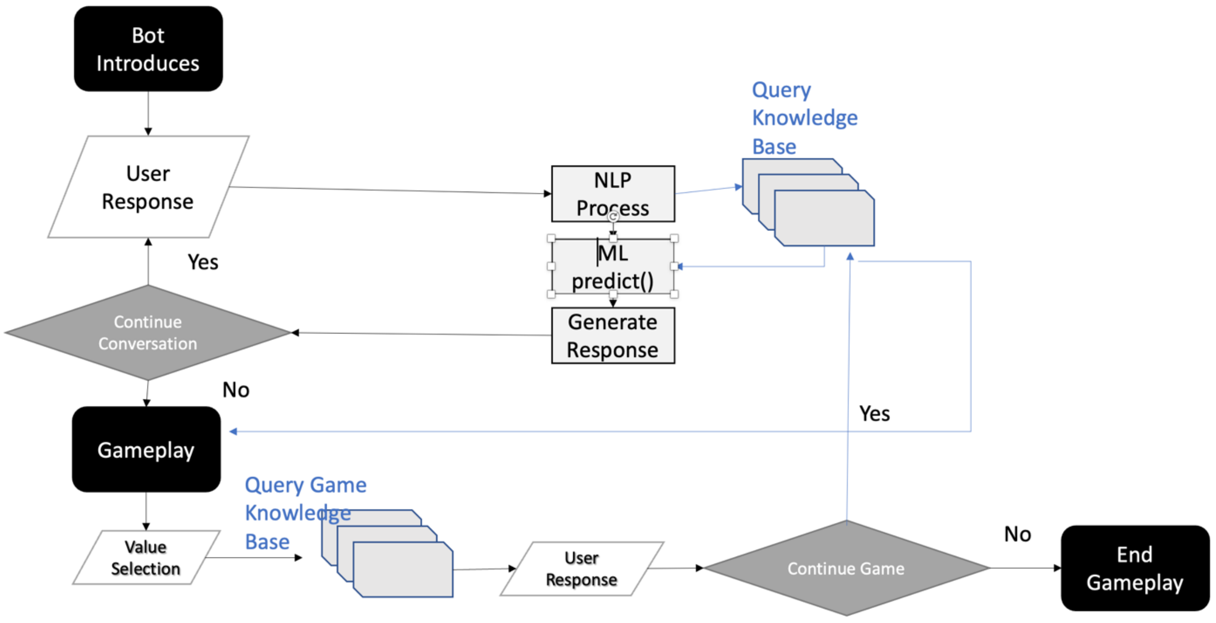
After, having two knowledge bases to query, the next step was to implement a process to modify user input, and match it to the data that had several phrases and key values. NLTK preprocessing was used to preprocess the data in the knowledge base and the user responses. Over tokenization and punctuation, I opted for the RegexTokinizer library that NLTK. This tokenizer, not only provides tokens on white space, but also removed punctuation. All tokens were lowered and with the use of NLTK WordNet Lemmatization, was used to break common words roots. Due to the fact that phrases and responses were short, stop words were not removed. Having the stop words included increased the number of words to compare with the user input.

Machine Learning Techniques.

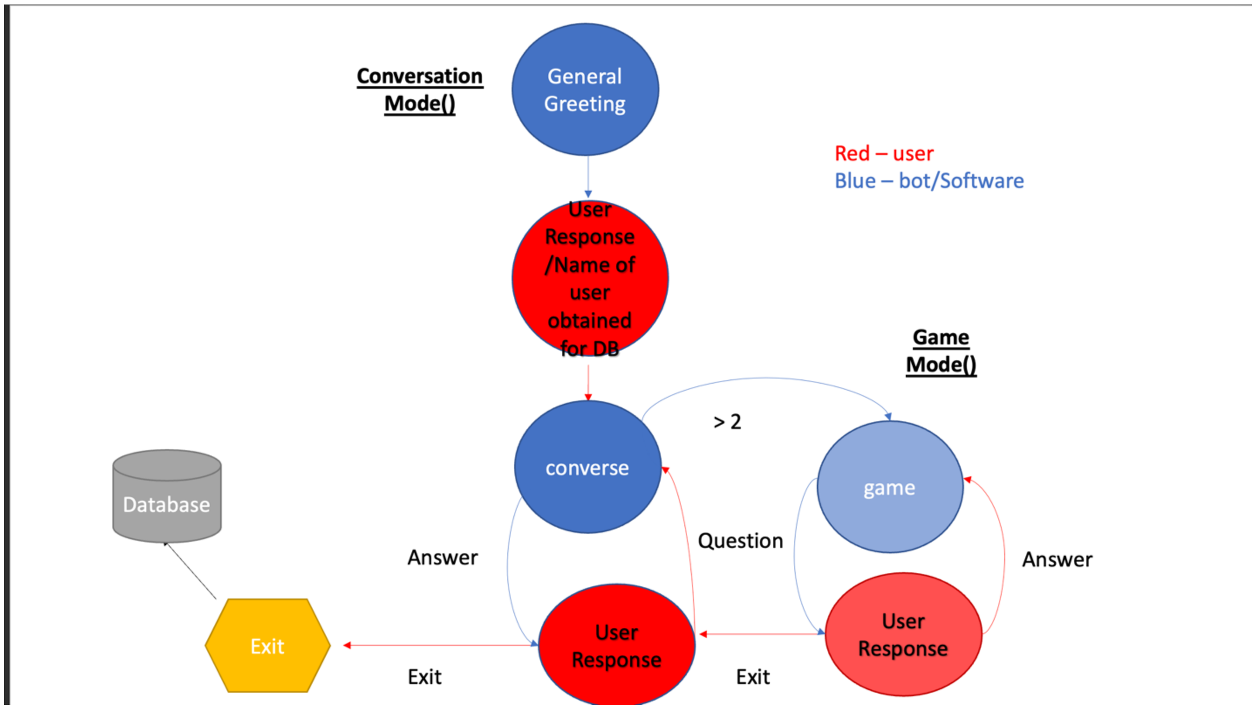
Originally, a deep neural network was going to be implemented into the original method in order for the prediction of the chatbot response. However, due to some conflicts with the M1 processor and the limitation on TensorFlow with this library, I opted for the machine learning algorithms in Sci-kit Learn. The first generation utilized a Naïve Bayes algorithm, but it was inconsistent. I looked into other Machine Learning models that are used in text categorization and I found the Logistic Regression and Stochastic Gradient Descent (SGD). Although I got mixed results, with Logistic Regression, in the opted Stochastic Gradient Descent. Prior to passing through the SGD model, all tokens were passed by through a count vectorizer. This produces a numerical representation of the tokes. The keys and the values were placed into a Pandas DataFrame, for ease of manipulation. Pandas has several functions that process and obtain information quickly from a large dataset, as is present in both our knowledge bases.

Logic and Dialogue Tree

The logic of the Chatbot was predominantly rules based to start and guide the conversation, however once in conversation mode, the ML model was the logic to provide Alex and interactive personality. The Jeopardy game, by nature, is rules based and which queries the Jeopardy knowledge base to produce the game environment. Please see the flow diagram of the logic utilized:



The over functionality is a state machine that “toggles” between “Conversation” Mode and “Game Mode”. As the names imply the conversation mode is an open-ended conversation of the Alex bot responding to User’s questions. Conversely, in Game mode, Alex provides all of the question and the user supplies the answers. The system was built by which Alex guides you back and forth between the modes by asking if the user would like to play a game to get into game mode, or asks the user if they still want to continue playing in order to exit or go back to conversation mode. Please see the conversation state diagram:



Appendix – Knowledge Bases

Intents Files:

This was a JSON file that had a dictionary named Intents. The internal keys included “tags’ by which a group of questions directed towards the chatbot was tagged. For instance, if the questions encompassed phrases such as “this is stupid” or “stupid chatbot”, the tag was “stupid”. The questions a user could provide were listed under ‘patterns and bot responses were listed under ‘responses”. The Knowledge base also had a context field in which all were blank, which was not utilized.

Jeopardy Knowledge Base:

The Jeopardy knowledge base was similar to the intents file, however had the keys of category, air date, question, value, answer, round, and show number. Show number and air date (the date the question was broadcast) were not used . The original intent was to support value, category, question and answer, however due to time constraints, only value, question and answer were queried.

Appendix – User Model

A data base and simple pickled dictionary. The database was an SQLite database that was intended for use with a Flask Webapp. This was created as the original intent of the chatbot was to be a web application.

Due to the time constraints, a pickled dictionary was made to hold the same information as the data base. The information contained in both includes the user Name, a system defined user id that is based on the Datetime information and a random generated number, the responses of the user, and the Jeopardy game score of the user.

This was incorporate into a dictionary as follows:

```
user = {
    "user_name": None,
    "user_id": None,
    "user_questions": [],
    "user_score": None,
    "setup_time": None

}
```

Evaluations

The evaluation was created with a Lambert Style Questioning that included the following survey.

From a Range of 1-5 where 1 = low, 2 = fair, 3 = okay, 4 = Good, 5 = Excellent

1.) Overall, how would you rate your overall experience with the Chatbot?
2.) Was the Chatbot entertaining, on a scale of 1-5?
3.) On a scale of 1-5, how many problems did you have?

Of the people used to test my Chatbot included Tracy Crumbaugh, a co-worker of mine for a non-biased aspect. My sister, who has championed me in obtaining my Master's Degree and is closer to my age. Finally, to have an older representative, my father was given an opportunity to attempt to play with Jeopardy.

Evaluations

<u>Tracy</u>

Responses

1. 3
2. 4
3. 1

I was a little surprised by the low answer to the amount of problems Tracy had, and this made a good point to evaluation on my Lambert Questionnaire. Tracy, I knew for a fact had problems with the Chatbot in the conversation mode. She confirmed that the Gameplay and game mode was fun and informative, however, she had problems in the conversation mode. It seemed at the time there was a variable error that was a critical error, in which one of her responses cause the system to crash. I realized that the Machine Learning model was not the best for text classification. I further complicated her experience by attempting to make changes to the code, which lead to further problems.

<u>Monica</u>

Responses
1.4
2.4
3. 4

Monica indicated that she did not have many problems but did not play too much due to the fact that she was not a fan of Jeopardy. She did think that it was "cute", with no indication what those entails, other than it was an interesting spin on Jeopardy. Monica was also critical of the UI, as it was too generic. However, she did provide good feedback to include a means by which if you get the question partially correct, you would get the full points. As in example, one question the answer was Huck Finn and she responded Huckle Berry Fin and got it wrong.

<u>Rafael:</u>

<u>Responses</u>
1.)4
2.) 5
3.) 4

My father did not get the full meaning of why you would want to "talk" to a computer when there are already good people to talk to. However, he had fun playing Jeopardy. It seems that the ML model was particularly not that good with him (however, I think I had him use the Linear Regression Model) and it was not fully trained). I wanted him to try with the SGD model, however time did not permit.