General Explanation of System and its Goals

This chatbot has a theme of TV Shows. The aims of the chatbot are to be able to have a conversation regarding a variety of tv shows and related conversation topics regarding tv shows, such as recommendations for new shows to watch, or talking about the user or alternatively, the chatbot’s favourite tv show. It utilises a rule based and similarity-based components to achieve this.

System Requirements

* The user should be able to greet and receive a response from the bot, using a greeting such as “Hello” and “How are you?”.
* The user should be able to type a response following AIML rules to receive an appropriate response as detailed in the AIML file. This will cover:
  + A response with a conditional word and then a personalised phrase afterward such as the AIML rule “Hi \*” allowing for “Hi mate”.
  + Some messages that will receive a random response. For example, having a variety of greetings to choose from.
  + Some information should be saved by the chatbot and can then be displayed back to the user when asked the appropriate question. In this case the user’s favourite tv show.
  + The chatbot will ask questions back at the user for example asking, “How are you?”.
* The user should be able to use a specific phrase to ask for an image to be displayed by the chatbot. The image will relate to the chatbot’s favourite tv show.
* The user should be able to use a specific phrase to search Wikipedia using the Wikipedia API. This can be used to search for the official summary of any tv shows.
* If none of the AIML rules are followed, the chatbot should defer to the similarity-based component and provide a response from the CSV file.
* The similarity-based component should allow the user to enter a similar question to the ones held in the CSV file and the chatbot should still retrieve the appropriate answer.

Employed AI Techniques

**Rule-based:** A rule-based component system, in the use-case of a chatbot, utilises a set of rules to make a choice. Information provided such as user input is checked against the rules, and the rule it matches will return a response relevant to the input and the mentioned rule.

**Similarity-Based:** A similarity-based component systems main aim is to use a similarity function to measure how alike two objects are. In the case of the chatbot, these objects are the user’s input, and some predefined questions that also have assigned answers. This chatbot utilises term frequency-inverse document frequency (tf-idf) coupled with cosine similarity to achieve this. Tf-idf is a statistical tool that can show the importance of a word within some information. The value increases proportionally to the frequency of the word and is balanced by the number documents that contain the word. Stop words can be filtered as they are extremely common and can disrupt tf-idf due to the weighting factor. The cosine similarity measures the “similarity between two non-zero vectors of an inner product space”. Together, a value will be found for the user’s input and predefined sentences, the closest matching of which will be deemed the most similar and retrieve the appropriate answer.

Program Explanation

The program contains several functions and initialisation variables. They are as follows:

The initialisations create the kernel object through the bootstrap method. It utilises a brain file as an argument, which is optional. From the NLTK library, the Lemmatiser, used for tokenisation, is instantiated. Additionally, for removePunctDict, a dictionary is created containing all the punctuation within string.punctuation with Unicode value None.

The readCSV() function opens the CSV file and initialises the CSV reader. Questions and answers lists are created, and subsequently the reader will add the questions and answers from the CSV file to their respective lists. The questions will then be made lower case and tokenised before being returned.

The lemmaTokens(tokens) function uses the lemma object and lemmatises each token in tokens.

The lemmaCleaned(data) function removes punctuation from the lemmatised words, makes them lower case and then word\_tokenises them.

The responseCSV(userResponse, sentTokens, ansCSV) function handles the chatbot to check for similarity based inputs. A bag of words is created using TfidfVectorizer. It is fed the cleaned and lemmatised tokens along with some stop words to ignore. Subsequently fit\_transform() applied. Then cosine similarity found between 2 arguments, then sorting and flattening of data. If reqTfidf is 0 then there is no similarity and appropriate response is returned. Otherwise the appropriate response from the CSV is returned.

The main function introduces the bot and starts a loop to keep getting user input after each response from the chat bot. User input will be taken, and then the response agent uses the AIML file to find a potential answer. If the first character of an answer is a “#” then will split the answer into different parameters, first being the command index and the second being the actual answer/information. The command is converted to an integer and checked via if statements. 0 returns a standard response; 1 uses a try block to search Wikipedia using its API and return a summary; 2 will display an image using matplotlib; and 3 will handle the similarity-based component. The similarity based component calls readCSV(), appends the user’s input to the list of sent and word tokens, then converts the set of word tokens to a list. Finally it will print the chatbots response, calling responseCSV() in the process, and then remove the users input from the sent tokens. Lastly, if there is np “#” it will just return the normal response from the AIML.