Building an ESL Chatbot

Third Capstone Project Proposal
Robert Walker

The Objective

The goal of this project is to create a chatbot that can carry on a conversation with a user who is learning English in order to help the user practice. The bot should be able to ask questions that elicit responses from the user, and then correct any spelling or grammar mistakes in the user's answer. The bot should also respond to the user's answers and ask follow-up questions in order to continue the conversation.

The Data

The dataset for this project will be created by scraping ESL dialogues from the following website: https://www.eslfast.com, which contains simple conversations on a wide variety of topics. Those dialogues will be used to train the model, however the model also needs a list of starter questions to begin the conversations. Such a list can be found at this website: http://iteslj.org/questions/getting.html, which will also be scraped.

Methodology

Web Scraping

As mentioned above, the first stage of this project will be to use web scraping to acquire data needed to train the chatbot. The data will be scraped from the two websites using the requests and BeautifulSoup libraries. First, the requests library is used to make a request to each website to obtain the HTML source code. Once the source code is obtained, it can be parsed with BeautifulSoup. After that the content can be extracted by searching for tags that contain it. For the first website (https://www.eslfast.com/easydialogs/), this will be accomplished by writing a function that can be used in a loop to scrape many pages of dialogues in succession. For the second website, there is only one page to scrape, so there is no need to use a loop. In both cases the results will be returned as a list, which will be converted into a pandas series. Afterward, both of the series will be saved as csv files.

Building the Chatbot

The next stage will be to actually build the chatbot. For this project, this will be done using Chatterbot, a great Python library designed to create chatbots that is relatively easy to use. A

Chatterbot model uses a logic adapter to select a response to a given input. Most of the logic adapters are quite simple. For example, the Time Logic Adapter simply returns the statement “The current time is…” plus the local time. The chatbot for this project will make use of the Best Match Adapter, which finds the closest match to the input statement and returns a response based on a list of known responses. Two functions are involved in this process. First, a statement comparison function is used to find the closest match to the input, and then a response selection function is used to select an option from the list of possible responses. In both cases there are multiple options to choose from. The statement comparison functions utilize different string similarity metrics, including Levenshtein Distance, Jaccard Similarity, and Spacy Similarity (which makes use of the Spacy library). The different response selection methods include “first”, “most frequent”, and “random”. Various combinations of these functions will be explored in the development of a final chatbot.

There is also a parameter for “similarity threshold”, which determines how similar statements must be in order to be considered a match. Finally, there is a default response parameter which can be a string or a list of strings. A default response is returned when the model cannot find a match to the input statement that exceeds the similarity threshold. The chatbot for this project will simply respond with a random starter question anytime it gets confused. This will keep the conversation moving, which is important in ESL.

Training the Chatbot

The chatbot can then be trained on sample dialogues from which it will select its response statements. The Chatterbot corpus (which includes multiple languages), is a great place to start. After the chatbot is trained on the Chatterbot English corpus, it will be trained on the dataset of ESL dialogues that were scraped from the internet for this purpose. At this point the chatbot will be able to respond to input statements by selecting the best response from the training material.

The Finished Product

The next step is to create instances of both PyDictionary, which returns definitions of words passed to it, as well as GingerIt, which performs spelling and grammar corrections. Both of these will be incorporated into the final product, so that users can have their input statements corrected if necessary, and be able to ask the definitions of unknown words. At this point, all of the different parts can be put together in a final algorithm. That chatbot will run inside of a while loop. If the user input is free of spelling or grammar errors, it will simply return a response. If a mistake is detected, it will return a corrected version of the input statement, followed by a response. If the user asks what a word means, a definition will be returned. The while loop will run until some text (like “bye”) is entered that breaks the loop.