Wilson Ding

Keaton Khonsari

CS 4301.001

Karen Mazidi

NLP Semester Project

Phase 3: Report

The KUWTK chatbot contains several subroutines to scrape websites, create a knowledge base, and finally runs the chatbot itself. This report will detail the specifics regarding the chatbot and its implementation, how each of the subroutines works, and an evaluation of the chatbot.

The chatbot contains the “urls”, “text”, “kb”, and “chat” subroutines, which are run in that order. Through the execution of all subroutines, the chatbot utilizes web scraping, input blacklisting/filtering, text extraction, input sanitization, word tokenization, stopwords, sentence tokenization, polarity analysis, and object serialization. The specifics of how each of these natural language processing techniques are used will be detailed below.

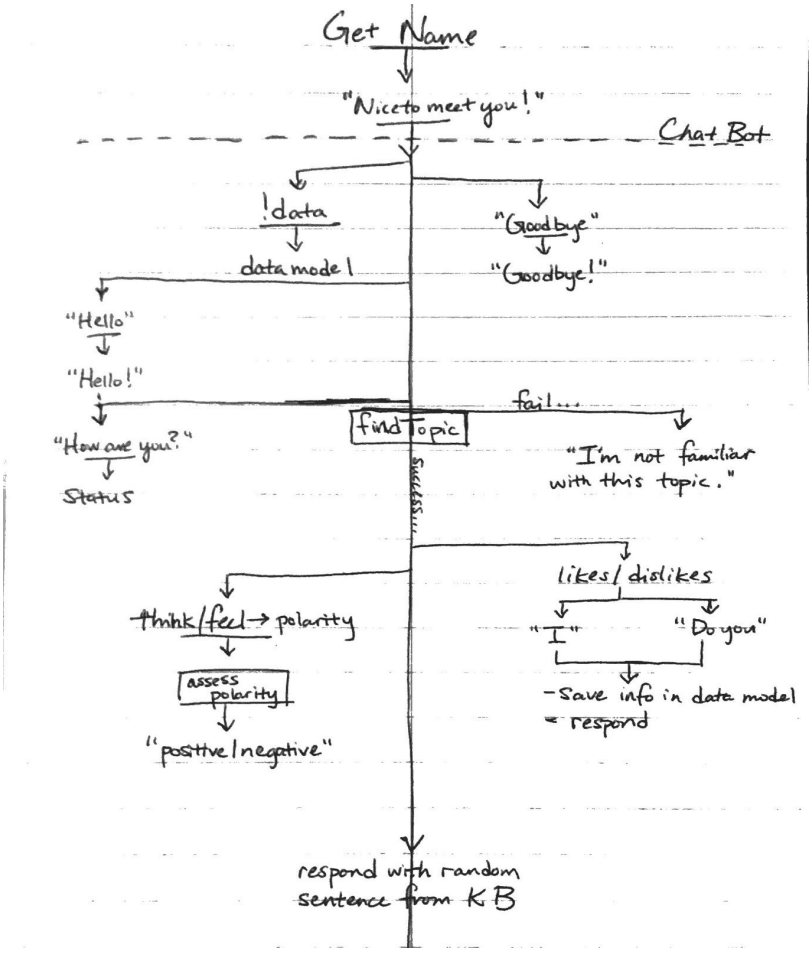
First, the “urls” subroutine attempts to create a list of relevant URLs for the purpose of later scraping to create a knowledge base. This is accomplished by first using web scraping via Python’s Beautiful Soup library, to scrape the beginning wikipedia URL for other URLs from the website’s HTML. These URLs are then filtered via input blacklisting, to remove inputs which can not be scraped, or does not contain relevant information. This remaining URLs are written to a file “urls.txt” in the working directory.

The next subroutine is “text”, which attempts to scrape each of the URLs created in the “urls” subroutine, getting the raw text, and saving the visible text for processing. This is done by visiting each of the URLs listed in “urls.txt”, extracting all raw text using Beautiful Soup. The text is then filtered, such that only visible text is saved. Each URL is saved under a “texts/” folder within the working directory.

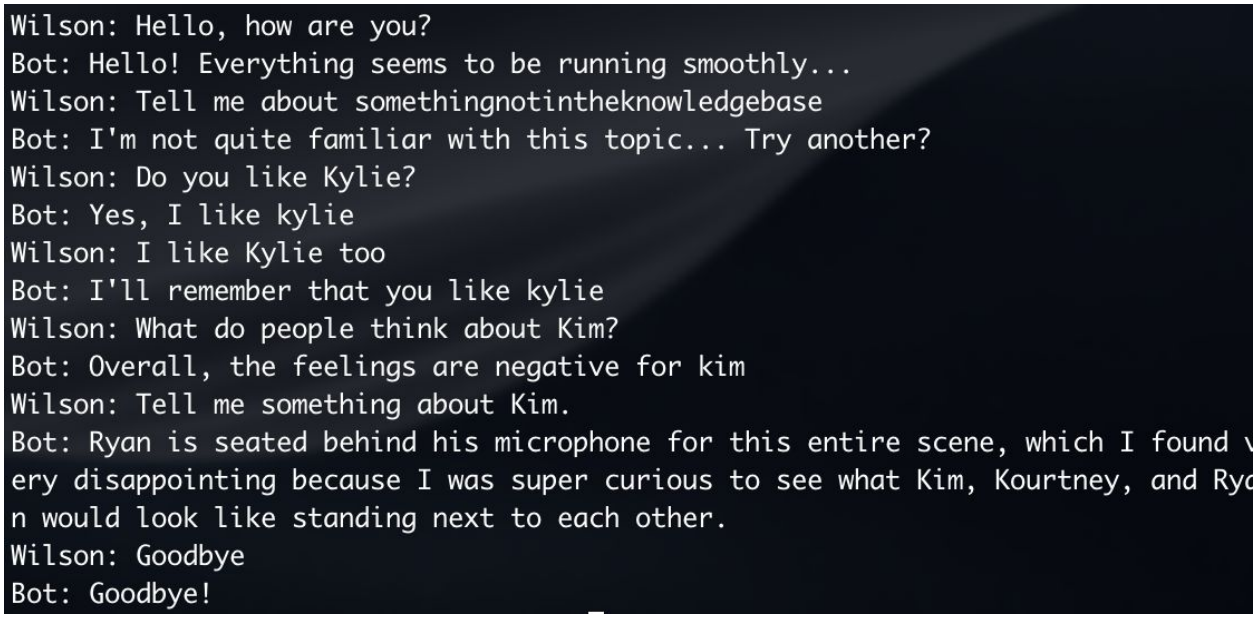
The next subroutine is “kb”, which attempts to sanitize the scraped text from the “text” subroutine, and then the top fifteen important topics/terms are determined, from which a knowledge base is created. This is accomplished by visiting each of the texts listed in the “texts/” folder, then feeding the scraped raw text through some input sanitization, where the text is “sanitized” by removal of punctuation, then normalized to remove excess spacing and converted to lowercase. After the text is sanitized, we then try to find a list of the top fifteen important topics and terms. We accomplish this through the usage of word tokenization to get all of the tokens from each sentence. We then import a set of English stopwords, in order to remove the stopwords from our list of top fifteen important topics and terms. Finally, we sort the list by frequency, and the top fifteen most frequent terms becomes our list of important topics. We then attempt to save all of the sanitized sentences into a “sanitized_texts/” folder within the working directory. As part of this saving process, we are able to read through each of the sentences and add them to our knowledge base if they are relevant to the top fifteen terms we discovered earlier. We do this through sentence tokenization, which we are then able to check for relevant terms. If an important term is found, it is added to the knowledge base, such that the key is each of the important terms, and the values are a list of all relevant sentences that contain the term. After all sanitized texts are saved, the knowledge base and terms are

saved in the working directory as “chatbot_kb.pkl” and “chatbot_terms.pkl”, using object serialization via Python’s pickle library. This allows us to persist the object data for the knowledge base and important terms between runs.

Finally, the last subroutine is “chat”, which takes the serialized knowledge base and list of important terms, and runs a chat bot to interact with the user, providing information on the fifteen important terms. The chat bot also uses polarity analysis via NLP’s SentiWordNet, in order to analyze the knowledge base for the sentiment of each of the fifteen important topics in the knowledge base. The specifics of the chatbot logic are shown below in the diagram of the chatbot dialog tree.



Below is some sample dialog interactions with the chatbot, in accordance to the dialog tree above:



In evaluation of the chatbot as described above, it has the strengths of running quickly (due to not using machine learning and needing to load training models), as well as being "realistic" in the chat messages. One thing we emphasized with this chatbot was the usage of "normal" conversational English, in an attempt to make the conversation more realistic. Furthermore, the bot seems like a real person because it has its own opinions of each of the Kardashians. On the other hand, the chatbot is weak when attempting to discuss specific facts about each person, where there is a single right answer. This is due to the lack of human training/improvement to the chatbot. Furthermore, the chatbot is unable to keep track of context, and previously discussed topics. This prevents the chatbot from being able to respond to multiple lines of conversation regarding the same topic without referring to the topic's name directly each line.

Appendix

Knowledge Base (with samples):

- Knowledge base is made up of a dictionary where the key is each of the fifteen important terms, and the value is a list of relevant sentences.
- Ex: {'kim': ['Kanye West will produce Kim Kardashian's debut album over the course of two decades.', ..., 'Kim and her sisters have designed jewellery and clothing lines for various companies; they put their name behind a sunless tanner, Kardashian Glamour Tan; and they own a clothing boutique.'], ...}

User Model:

- Upon compiling the chatbot program, before initiating a dialog exchange with the chatbot, the user is asked to enter a username they'd like to go by.
- Once the user finishes entering in the name they would like to go by, the program then checks the current working directory to see if if a pickle file titled the entered username the user had input.
- I.e., the user enters "Wilson" as their username, the program then looks for a wilson.pkl file, which would contain that user's saved preferences (likes and dislikes) from any previous dialog exchanges that user would have had previously with the chatbot.
- If this is the user's first time, it will then automatically create a pickle file named after the user, and will save any preferences that the chatbot discovers during the dialog exchange with that user.

- When the command ‘!data’ is entered during a dialog exchange, it will return that current user’s preferences that is saved in their pkl file as so:

```
Keaton: Hello
Bot: Hello!
Keaton: How are you?
Bot: Everything seems to be running smoothly...
Keaton: I like khloe
Bot: I'll remember that you like khloe
Keaton: !data
Bot: User data for keaton: {'dislikes': 'kylie', 'likes': ['caitlyn', 'jenner', 'kylie', 'khloe']}
```