Jackson Ginn

CSCE 240 Prog 6 Report

Requirement

We were instructed to create a chatbot that could provide information about a representative of our choosing. The chatbot had to be able to handle any query, and it had to be able to provide specific information to the user when given certain queries. The program took data from the representative's website to process and give to a user. The program also had to store statistics of interactions and make them accessible to users.

Specification

I created a chatbot to answer questions about District 83's representative, Bill Hixon. I copied the data directly from https://www.scstatehouse.gov/member.php?code=849715808, but I did not use curl or any web request program to access it. I then stored this data in InfoSets, which are a class I created. These InfoSets are serialized and used in other programming assignments. The program checks a user's utterance against the list of supported queries using Levenshtein distance, and if a query is within a certain threshold of the utterance, its associated data is printed. The program stores the chatlogs in the data subdirectory, and statistics for each session are kept in a csv called chat_statistics.

Development Highlights

The only class I created besides the Chatbot class (which contains the main method) is InfoSet. An InfoSet has a String title and String[] facts, so it stores relevant data about the representative.

The class can easily provide its title and its facts. The program has an array of InfoSets to store all of the data, which allows for easy access when a query is called. I tested the program by first running all of the supported queries. I then tried variations of those queries to make sure it could handle typos and close matches. This involved adjusting the tolerance for the Levenshtein distance to avoid excluding some matches while not including false matches. This fine-tuning was one of the most difficult parts of the program, as it was a delicate balance to find. The implementation of the distance measurement was also difficult, as it required implementing another library from Apache Commons and learning how to use it.

Reuse

To make my code reusable, I avoided hardcoding anything specifically related to my representative. Because of this, my program can create a chatbot for any representative provided text copied from that representative's website. My code is generalized to where only minor modifications are needed to reconfigure it for a new chatbot. No one used any of my code, and I did not use any other students' code. The only code that I used and did not write was the package for Levenshtein distance from Apache Commons. I did not face any challenges in making my code reusable. In fact, I found it easier to generalize my program rather than hardcode things for my representative.

Future Work

The main thing that would make my chatbot more useful would be to expand the number of supported queries. This would allow for users to learn more about the representative by making some facts more easily accessible. Another potential new feature would be for the bot to answer basic conversational questions like “how are you”. This could engage users more and make the chatbot feel more personable. As for code changes, the input file from the website will need to be updated with new facts about the representative if they are added. In the event that the representative is not re-elected, then the chatbot would also need to be reconfigured. However, since my code is reusable this would not be very difficult.