Cosc 310 Individual Project Jesse Plamondon

Repo: https://github.com/jesseplamondon/310IndividualProject.git

For my individual project, which has been done in Java, the user takes on the role of an interviewer and the agent takes on the role of an interviewee for a software engineering job. The user prompts the agent by asking questions related to their history, experience, etc. and the agent responds with a relevant answer. The software developed utilizes Named Entity Recognition, Coreference Resolution, SpellCheck, Wikipedia API (jwiki), Language Translation and POS Tagging to identify various structures within user input phrases to respond appropriately. The software utilizes a minimalistic User Interface which allows users to easily enter interview-based phrases and receive responses that are cohesive to an interviewee.

Added Features

Google Translate API

If the user inputs text in a different language, then the program will translate the text to english and use this string to compare with the database. Otherwise, the system would not be able to detect any keywords and would not be able to function properly if the user spoke a language other than english. The system then translates the response to the language the user input with and outputs it to the user.

Wikipedia API

This functionality is built off of the named entity recognition and, if the answer output by the system contains an entity recognizable by wikipedia, then it will add information on that entity that has been collected from wikipedia to the end of the output phrase. This allows the system to show an understanding of entities used in phrases, which is useful for an interviewer type bot in which demonstrating knowledge is central.

Features

Misspelled Word matcher

If a potential keyword doesn't match any known keywords, then the program will try to match it against known words. If the misspelled word nearly matches a keyword, the program will recognise it as the keyword.

POS Tagging
Without the POS tagger, the program must loop through every word in the input string until it finds a matching keyword. By adding the POS tagger, the number of words the program must search is reduced. The POS tagger takes the input string, tags each word, and returns a list of potential keywords based on their tag. So instead of searching the entire string, a list of two to three words is searched instead.

Named Entity Recognition
The named entity recognition improves the responses our chatbot gives to particular questions without those responses needing to be hardcoded. The named entity recognition allows the program to recognize names, and use those names in its responses. For example, if the interviewer asks if the chatbot has ever traveled to a particular place, the chatbot can recognize the place name and use that name in its response for a more natural sounding conversation.

Coreference Resolution
The coreference toolkit enhances our chatbot by improving the flow of conversation for the interviewer. Instead of explicitly stating the topic each time, the chatbot is now able to match a previous topic to the current conversation. For example, the interviewer can ask the chatbot about work experience, and then ask the chatbot to talk more about it. Even though a particular topic is not named, the chatbot can recognize what the topic is from previous conversation.

Compiling and Running

CSV Path Change
After getting all of the files from the GitHub repository the CSVs location must be changed. This can either be absolute or relative and is found within ChatBot.java in the search function.

JavaFX
After this JavaFX must be installed onto the machine.

VSCode:
* Update VSCode to 1.49.3 or above.
* Install JavaFX support extension in VSCode.

* Download and install open JDK and configure Java runtime in VSCode (may need to add to system variables).
* Download JavaFX SDK, extract, and copy the path of lib folder.
* Reference all jar files within JavaFX SDK to the Java project.
* Edit run configuration within launch.json by adding:
`"vmArgs": "--module-path {Your path}/javafx-sdk-11.0.2/lib --add-modules javafx.controls,javafx.fxml", `

Eclipse:
* Follow this tutorial: https://www.youtube.com/watch?v=bC4XB6JAaoU

CoreNLP

After setting up JavaFX. The next step is to properly reference the Stanford CoreNLP jars.

Generalized method:
* Download the CoreNLP zip file from: https://stanfordnlp.github.io/CoreNLP/ and extract
* Reference all of the jars to the classpath of the Java project

Jwiki

After including the CoreNLP API, include the jwiki jar file found in this repository into the classpath of the Java Project.