Critical Reflection - Omar Choudhry [sc20osc]

What went well with this project?

I managed to create a chat bot that met all the main requirements set in the planning report and some of the more advanced objectives I hoped to achieve. Simply, I managed to get my chatbot, UniBot, to store the user input, clean it by removing punctuation, symbols, digits, redundant spaces and changing character case. I managed to create a process to scan the input for keywords or key phrases and search the knowledge base for a response, based on the input and then output the response. I managed to do all of this whilst storing the entire conversation in a file containing conversation logs.

As for the advanced objectives, I managed to deal with null inputs, primarily scan the input if the user wants to quit, and one of the best things I achieved was the mechanism I implemented to control repetitions. Repetitions are split into 3 categories, the first being null input repetitions which was simple, the second being exact same inputs, and the third, being the most complex, similar inputs that would be classed as a repetition.

In short as I mentioned in the planning report, the user may try to trick the chatbot by repeating the previous input with a slight modification, which needs to be counted towards a repetition. And so, I created an efficient process that scans the input for similarities.

What was the hardest part of this project?

The hardest part of this project was testing the behaviour of the chatbot. Considering the process calculating was so much more complex than a simple expected value, it meant that using unit tests was very difficult. Essentially, since the response change with every input it becomes difficult to keep track of what is going on and the program needed to test would be the same as the one implemented. This means that the test would always pass since the algorithm is the same for the testing and the main program.

This also meant testing the output of the conversation log file also had to be done manually. Since unit testing cannot be used “easily” to test the specific responses, it means that the only thing you can test would be the file format, but again this become difficult depending on the conversations and if some were to fail or end abruptly it would cause a problem with the log file. Therefore, I had to implement the conversation logs towards the end of the project.

Why, and what will you do to address this for the future?

Even though it is difficult, testing the response and input mechanism is only possible with conversation logs. Instead of constantly appending conversations, I could have only stored the most recent conversation (I preferred not to do this to make the logs more meaningful) in the file and compare the outputs of both files. To do this effectively I would only keep a maximum of 2 responses in which case I can control bot response repetitions. For the future, I would assign more time to testing to be able to catch this problem earlier than later to figure out a solution.

How could I improve?

As I mentioned previously, I had set some more advanced objectives to complete. To improve I would go back and try to complete those. This includes, using a flat file to store the knowledge database. This way adding to the knowledge would be very easy, and it would require an algorithm to analyse the file effectively. Others include adding an ability to understand multiple languages and create a leaning capability. Adding the ability to understand multiple languages is difficult considering it doesn’t even understand English, it simply responds if the input contains at least a keyword within the knowledge base. However, adding a simple algorithm to check for characters in other languages would help in at least responding with a message like “I don’t speak French!”.

As for the learning capability this becomes a lot more complex and due to the time and memory issues I couldn't implement this. Essentially I would create a memory mechanism for the chatbot that would note inputs that the user enters and try to make sense of if it is an adjective, verb or noun and create a response based on what the user responds with. An easy way would be to transpose the input. Sentence transposition is a basic way of conjugating the verbs of the input and change them from first person the second person or the reverse. The other thing would be adding the concept of keyword location and introducing context. Although some keywords can be found alone within the input, some cannot because otherwise the sentence wouldn't have any meaning, for example the key phrase "who is", should not be found alone since there is no meaning. As for context, the chatbot would store some information about the conversation, for example if the chatbot asks for the user's name, the chatbot should store this value and utilise it within responses.