String Comparison Chat Bot

# Problem Space

After seeing several examples online of people creating chat bots that (try to) speak like they would I decided it was something I’d like to try. The only problem was that most chat bots used machine learning, a subject I’m not versed in enough (yet) to attempt, so that got me thinking how I could create a similar bot without any machine learning. The bare-bones idea behind this was to create a program that had an extensive library of messages and responses. This way the computer could simply compare input strings to previous messages and pick the most appropriate response possible.

# Solution

The largest problem in this project was the data cleaning. Sifting through 4GB of JSON files from facebook would be tortuous task, almost inhumane. Cleaning the data required a skim analysis of the data to find as many potential problems in the data. The first noticeable problem was that the messages were recorded in reverse-chronological order. Thus, any conversation read would have to be fully reversed. The next step was a trivial lowercasing of all messages, removal of all punctuation, removal of web addresses and standardisation of elongated strings (e.g. loooool). The reason for this was to make string comparison easier. After the data was clean it was time to prepare it in a library format. To be done quickly the basic method was to take a message sent to me, concatenate all replies (until the next message from the sender) and create a pair. Once you have a list of all pairs then you can compare input strings with messages received to pick a reply. Obviously, this is very rarely going to make sense, but it’s a fun idea and has the potential to produce (somewhat) coherent sentences. One can compare strings in many ways; such as the Levenshtein distance which I’ve written before. In this case I decided that I wanted to write my own “algorithm”. It calculates a similarity score based on the two lengths of the string and the number of words in common (as a set). I thought this would be a good method based on the idea that messages of this length are likely to be more coherent when they have more words in common. Especially when it comes to niche topics. You can download your own Facebook messages history in JSON (Low-Quality) format and put the inbox directory in the project main folder. This way you can see how it works with your data. I would recommend only choosing the past couple years of messages as your online presence can often change over time (there may also be some old messages that are better forgotten). The program also contains some methods and classes for a different type of chatbot I worked on but it is obsolete.

# Example Conversation

This was my favourite conversation for the accuracy and the fact it almost is a full flowing conversation, for clarification the Hams Hame, Criterion and Quays are all pubs.