Siena Okuno

Soft Des 2017-2018 Fall

The Program of Inspiration (Sort of)

In this assignment, the main goal was to use text mining to create an interesting end product involving accessing information from the internet, parsing text and pickling it, and complex computational methods to characterize/compare the text. I decided to create an Inspiration Bot that gives interesting, semi-coherent blurbs to hopefully brighten its user's day. My program took the information from inspirational quotes on Twitter, using Markov's Analysis to mimic the quotes' structures and word choices.

To start off this project, I decided to work with pickling my data. Twitter's API rate limit is only 100 per hour, so it was very important to store my data instead of collecting it online every time my program was run. The key was discerning whether or not data was already cached, so to check for this I created an 'if not' loop in my Pickling.py file to check for the cache file and if it did not exist then use 50 API calls to search for inspirational quotes and gather them. From here, the information was then "unpickled" and used in my GetTweets.py file. This file implemented the pickling and unpickling functions and sorted through the raw information the search function gave me. The given information was a bit complex in nature but essentially within the main class type that was the entire search information there were .text attributes. I compiled the text attribute into a list and from there removed the "RT @ (username):" part and any tags at the end of the message. From here I joined the remaining quotes into a long string to be passed through the Markov's Analysis part of the code.

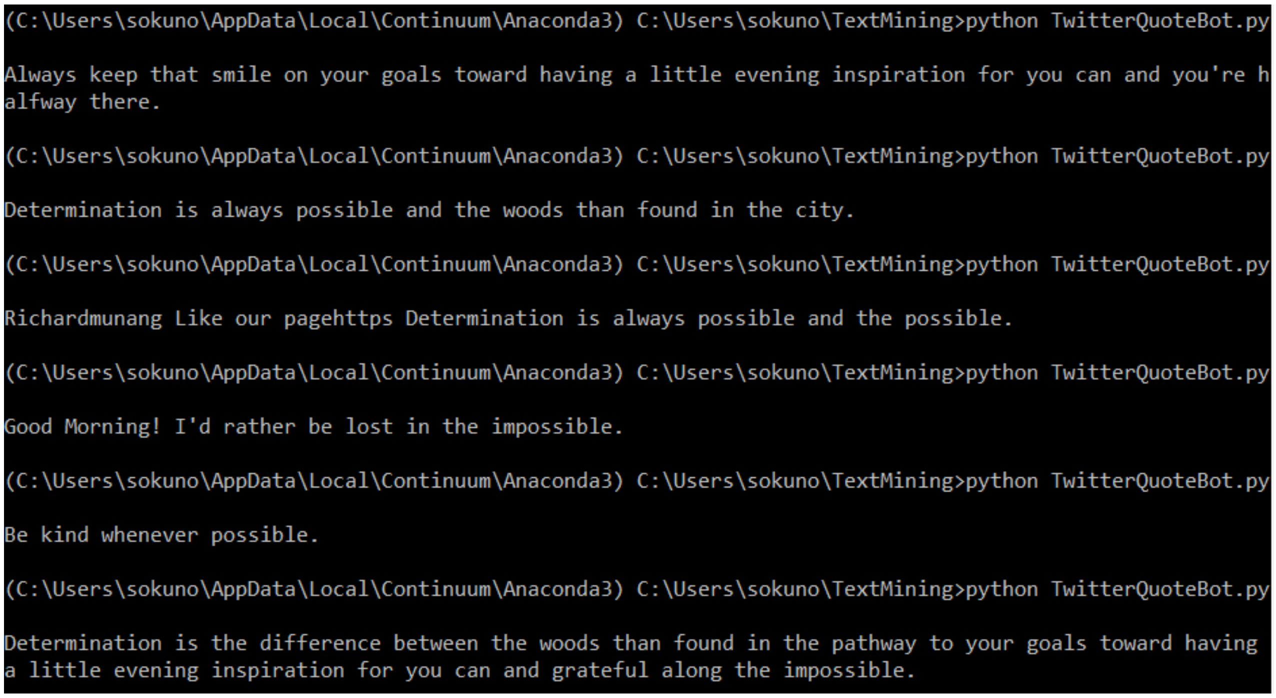
Markov's Analysis is a method used to predict the likely-hood of a future value based on its current and past values. This translates into word prediction and sentence generation in this case. The code I used[1] first standardized the case of every letter (besides 'I') and then began mapping out the possible word choices. First, addItemToTempMapping() works to relate words to their histories, as in what sentence-word relationships were given and how a new word is incorporated. The next function

[1] While I am very interested in this method of prediction, I did not write the code on my own but instead adapted one on the internet to work my previously existing functions.

builds onto this and incorporates a dictionary. After this, the next couple of functions use the map to predict each word and then a sentence, starting with a random word. This is all wrapped up in the final TwitterQuoteBot.py file that calls the functions, replaces any odd characters, and predicts a sentence.

Originally I looked at machine learning to teach my bot how to compose sentences. However, after some consideration I realized that it would take a long time to teach my program to speak well, so instead I went with language processing techniques. Again, not everything went as planned with my original idea of using the Natural Language Tool Kit (nltk) library. After trying to implement some code and researching the errors online I realized the library was buggy and not all functions worked in newer updates. I tried looking at some other codes found online but I couldn't get most of them to work. Instead I found a decent looking article and changed some pieces to fit the outputs I had from previous functions. In the end, I think this allowed me to understand the functions a bit more as they didn't use any fancy libraries that hid what the program was doing. However, I plan on experimenting more next semester to write my own Markov's Analysis program to better understand the mapping behind the function.

In the end, my program successfully could make different and new phrases out of the text I gave it. There isn't quite enough data to properly form totally unique sentences and some phrases tend to be repeated (i.e popular retweets like "Be kind whenever possible"). However, after increasing the cached data to include 100 tweets with the tag #inspirationalquotes, there is a noticeable difference in the uniqueness of the phrases even if sentence structure is off a bit. Here are some examples of the text it can generate:

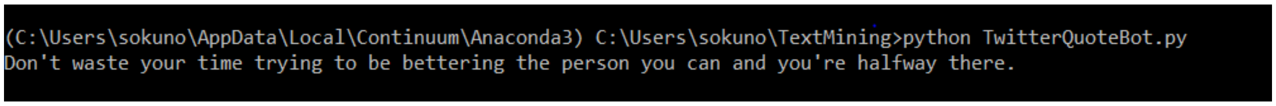


```
(C:\Users\sokuno\AppData\Local\Continuum\Anaconda3) C:\Users\sokuno\TextMining>python TwitterQuoteBot.py

Always keep that smile on your goals toward having a little evening inspiration for you can and you're h
alfway there.

(C:\Users\sokuno\AppData\Local\Continuum\Anaconda3) C:\Users\sokuno\TextMining>python TwitterQuoteBot.py

Determination is always possible and the woods than found in the city.

(C:\Users\sokuno\AppData\Local\Continuum\Anaconda3) C:\Users\sokuno\TextMining>python TwitterQuoteBot.py

Richardmunang Like our pagehttps Determination is always possible and the possible.

(C:\Users\sokuno\AppData\Local\Continuum\Anaconda3) C:\Users\sokuno\TextMining>python TwitterQuoteBot.py

Good Morning! I'd rather be lost in the impossible.

(C:\Users\sokuno\AppData\Local\Continuum\Anaconda3) C:\Users\sokuno\TextMining>python TwitterQuoteBot.py

Be kind whenever possible.

(C:\Users\sokuno\AppData\Local\Continuum\Anaconda3) C:\Users\sokuno\TextMining>python TwitterQuoteBot.py

Determination is the difference between the woods than found in the pathway to your goals toward having
a little evening inspiration for you can and grateful along the impossible.
```

One really interesting phenomenon I noticed while working with a smaller cache is that when phrases are repeated in the tweets, this is really reflected in the structure and word choice of the resulting sentence because the relationship as defined by Markov's mapping system is very strong, almost ensuring that certain words follow each other. This even occurred in entire quotes being returned by the program, the most recurring one being "Once you accomplish one goal, create another."

I think a larger cache would be very beneficial and would improve the results a lot because of the wide variety more quotes would provide. This would mean either editing the pickling function to open a file and add to it, having the GetTweets function be able to process more than one text file, or creating another function to combine files. In general though the project was a success in that it created mostly correct sentences that at least cheered me up with their odd phrases.

A bonus not-so-inspiring quote:



```
(C:\Users\sokuno\AppData\Local\Continuum\Anaconda3) C:\Users\sokuno\TextMining>python TwitterQuoteBot.py
Don't waste your time trying to be bettering the person you can and you're halfway there.
```