Yichen Jiang
Software Design: Project 3 Text Mining

Project Overview:

In this project, I use Conan O'Brien's twitter posts as my data source. By analyzing the language used by the famous talk show host, I want to generate random tweets with his language style by using markov analysis on his twitter messages. Through out the project, I want to understand the basic operations of python-twitter api and get better at handling data structures and text mining.

Implementation:

In order not to access to my twitter Api every time I test my code, all of Conan O'Brien's twitter posts are first stored in a local file with pickling. Pickling here is used for maintaining the raw data type collector.py collected from twitter. I noticed that twitter posts are usually followed by a series of #Hashtags and URLs which are things I don't want to keep track of when generating the random tweets. So I removed them by searching for unique character(s) such as # and https:/ and truncate them off the string. I also make all the words in lower case to create a more various map through markov analysis. Now that the messages are ready, I can start implementing the markov analysis.

Before coding the markov prefix map, I considered the appropriate types for it. Since markov need to map different prefix to suffixes, a dictionary is my best choice. Since this dictionary will be used throughout the project and dictionary is mutable, I decided to declare the markov map as a global variable. Now for the current_words, since it will contain multiple elements and will be stored as the key of the dictionary, declaring it as a tuple is the only way to get these jobs done. With these basic structures, I implemented the markov analysis by iterating through the messages, updating the current words and recording the word that follows. After the map is created, I used a while loop for identifying ending condition and generating funny tweets along the way. Similar results can also be obtained by recursion and I choose while loops because they make the code easier to read and not too much longer that recursion code in this case.

Results:

Currently, my code can generate a markov map with user-picked prefix length. Because a single word in markov analysis will make the text too random and a length over 3 will make the text the same as the original tweets (due to the small data set of tweets available), I sticked to the convention length of 2. The program can generate a tweet either based on user input of two letters or by complete random selection. The text generated in the second way is usually truncated in the middle so I choose the first way with the key words (i think) as a starter. 5 random tweets are demonstrated below(all in lower case):

i think of how that pen he's using once dreamed of a guy or girl who wants to meet me, for the next 72 hours i'll be standing next to the winner of the 2016 presidential election. you’ll be receiving the cleaning bill for america’s pants.

i think a lot of friends. conanmexico

i think of how that pen he's using once dreamed of a sears without alerting security.

i think a lot of job loss is attributed to automation. in fact, today i played fútbol with superstar oficialgio. here's my signature move: "running with limp hands." conanmexico

i think a lot in common.

Reflection:

My implementation of Markov analysis went pretty fluent. I really enjoyed implementing coding knowledge to solve a real life problem and being able to generate something that can make people laugh. Future improvement to the project can involve adding a flag to the first words of a sentence and capitalizing it while generating the tweet. This should help make the tweet more close to human language. Because the collection of Conan's tweets is not a very big data set, the output sometimes doesn't look random at all. Adding other celebrities' tweets can be a good solution to enlarging the data set and this isn't hard to achieve with my current code.