John Wen
Professor Steele
Software Design
10/11/17

Project Overview

The data source I used was Project Gutenberg, and the way I analyzed the text was through a word frequency counter as well as a function that counted the top N words used in the book. I hope to learn how to use Python to draw data from API sources and find meaningful and feasible ways to analyze data.

Implementation

First I had to create a function that cleans up the string of texts I drew from Project Gutenberg. This meant that I had to remove all the symbols from the string before turning the string into a list so I can count them with a dictionary. Similar to how the histogram function in class counted the letters, I was able to count the words with the .get function. Once I had the count of words, I was able to create a function that returned a tuple with the highest word frequency and its respective. From there I created a function that iteratively gave the top N values from the dictionary of words.

The second part of implementing the data analysis, I wanted to try use the Markov analysis to build a text. At first I tried using a dictionary for every word in the text and within that dictionary there will be all the suffixes (as keys) and their appearance counts(as values). From there I tried to return suffixes based off of a proportional probability of suffix appearances but this approach proved to be too difficult. I had to strip the proportionality aspect and return suffixes at equal chance. I first had to build a dictionary with every key value in the text. From there I had to use a while loop to append each suffix to a list in the dictionary values. I wasn't able to proportionally analyze how each word appears but I could generate a sentence with random suffixes at an equal chance.

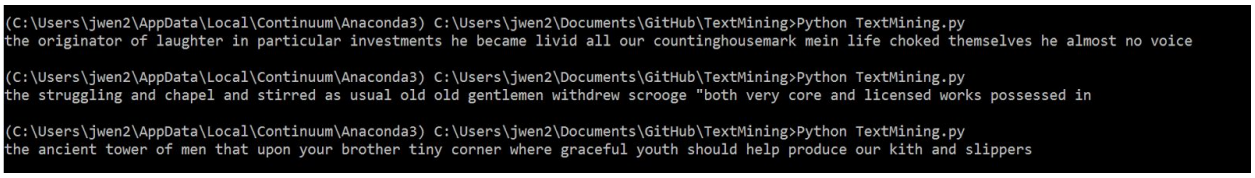
Results (Also Uploaded Excel Sheet)

A Tale of Two Cities	Great Expectations	A Christmas Carol	Oliver Twist	David Copperfield
Total Words: 138,846	**Total Words: 187,406**	**Total Words: 31,544**	**Total Words: 160,979**	**Total Words: 357,833**
The 5.84%	The 4.39%	The 5.48%	The 6.06%	The 3.79%
And 3.53%	And 3.64%	And 3.44%	And 3.37%	I 3.37%
Of 2.96%	I 3.17%	Of 2.46%	Of 2.47%	And 3.28%
A 2.52%	To 2.71%	A 2.45%	A 2.45%	To 2.89%
To 2.12%	Of 2.41%	To 2.32%	To 2.35%	Of 2.44$

A Tale of Two Cities	Great Expectations	A Christmas Carol	Oliver Twist	David Copperfield
Total Unique Words: 12,327	**Total Unique Words: 13,659**	**Total Unique Words: 5,373**	**Total Unique Words: 11,324**	**Total Unique Words: 19,902**
Unique Word Percentage: 8.87%	**Unique Word Percentage: 7.28%**	**Unique Word Percentage: 17.03%**	**Unique Word Percentage: 7.03%**	**Unique Word Percentage: 5.56%**

The most surprising part is that, the most frequently used words are very mundane words and their usage percent is pretty consistent across the board hovering at 4% - 6%. As for unique words used, they were also consistent around 5 - 7% with A Christmas Carol hovering at 17%. This is probably because ACC is a shorter novel (or play) and therefore have less of a chance of repeating words. In this case a unique word is a word that only appears once in the text.

As for my Markov synthesizer, I ran it three times for A Christmas Carol creating sentences with the length 20 starting with the word ‘The’ because it was the most frequent and the results are posted below.



Reflection

From a process perspective, I think the project was perfectly scoped. At first it was quite intimidating with all the texts and mining all the API’s from different websites but once I honed in on one such as the the Gutenberg, it became very possible to break it down step by step. Once one function was made, I was able to build onto the next and slowly learn things as I went. I became much more comfortable with using dictionaries after going through this entire process.