Project Overview:

In this project I used project gutenburg and two other sources to get texts of novels. Then I hoped to create some summary statistics of the following:

Summary statistics I want for each text:

Words:

- Total number of words
- num of words used
- most frequent 100 words and their frequency

Sentences:

- How many sentences there are (roughly)
- The average sentence length (in characters is easy)?

Sentiment:

- The average sentiment value of all sentences
- The sentiment value of whole text

Similarity:

- compare one text to all the other texts

Some summary statistics graphs that would be cool to have:

- plot of the sentiment of each sentence (frequency vs value)
- plot of length of sentences (frequency vs values)

I was able to do all of the above and write summaries to files which was something I thought was cool and useful.

Implementation:

After writing some pieces of code and debugging, I decided to write everything into functions because it makes it much easier to read and only execute codes I want. Furthermore, it makes everything look very organized. The only part not in functions is the beginning where I load my pickle files into list of string texts and create another lists of filenames which I later use to write files with.

My functions consists of two basic functions “Str_to_words”, “Str_to_sentences” that convert a text in to list of words and list of sentences. Then, I have text processing functions “Histogram”, “most_frequent (words)”, “sentiment_analysis” and “compare_similarity”. These uses the lists from the basic functions as well as output from each other to extract summary data from the texts. Then I have some functions that plot histograms from the summary data and also some functions that write summary text files into the directory.

The conversion between my data structures- strings, lists, dictionaries and tuples are quite straightforward inside each function. List comprehension helped in a lot of the cases.

One choice I had to make was to use two functions for plotting different histograms, where the code structure was the same for both. I could have minimize repeating code by using a single function to plot a histogram, but I decided to write two because there would be too many specific arguments of a figure to pass to a single function, like axis labels and titles.

Results:

The most interesting part!

Summary data in a text file directly created by my program:

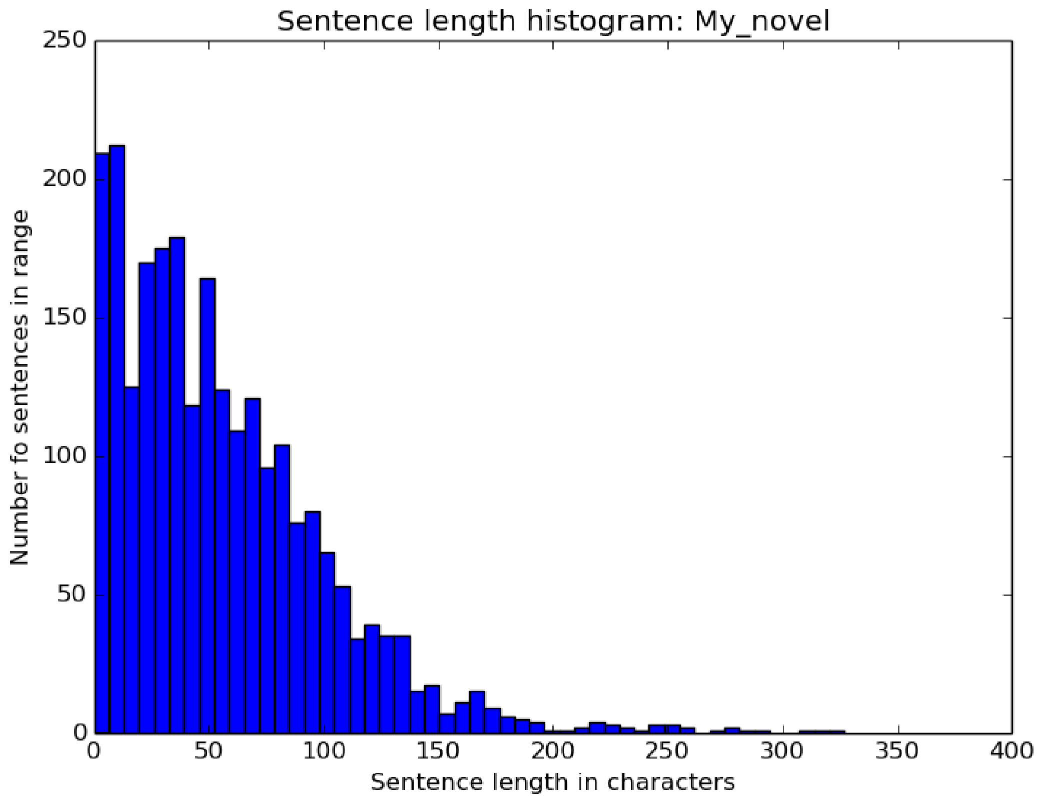
```
This text is Bronte-Wuthering_heights

This text contains 115911 words

The author uses 10219 Unique words

The most frequent 100 words are:

[(and, 4693), (the, 4551), (i, 3531), (to, 3476), (a, 2301), (of, 2220), (he, 1922), (you, 1712),
(her, 1544), (in, 1459), (his, 1419), (it, 1284), (she, 1269), (that, 1188), (was, 1124), (my,
1098), (me, 1047), (not, 932), (as, 931), (him, 917), (for, 836), (on, 809), (with, 804), (at, 783),
(be, 724), (had, 687), (but, 673), (is, 646), (have, 629), (from, 485), (by, 451), (would, 442), (if,
440), (heathcliff, 413), (your, 404), (no, 384), (said, 368), (so, 357), (were, 351), (linton, 340),
(catherine, 333), (an, 317), (we, 311), (mr, 309), (when, 307), (or, 307), (out, 305), (what, 301),
(are, 295), (this, 290), (they, 283), (then, 283), (up, 282), (there, 282), (could, 279), (all, 279),
(one, 276), (into, 273), (do, 247), (should, 246), (did, 240), (will, 236), (them, 229), (now, 224),
(been, 221), (how, 220), (more, 217), (than, 199), (shall, 193), (come, 187), ("im", 186), (must,
184), (master, 184), (which, 183), ("ill", 183), (us, 179), (never, 179), (little, 178), (go, 175),
("dont", 175), (see, 170), (down, 168), (has, 165), (hareton, 164), (after, 164), (over, 160), (its,
158), (about, 157), (answered, 155), (some, 154), (before, 154), (like, 153), (who, 150), (till,
148), (their, 146), (only, 144), (any, 140), (let, 138), (too, 134), (again, 134)]

There are roughly 8703 sentences

The average sentence contains 78 Characters

The average sentiment value of all sentences is 0.046, 0.546

Sentiment is positivity from -1 to 1 and subjectivity from 0 to 1

The sentiment for the whole text is 0.049, 0.508
```

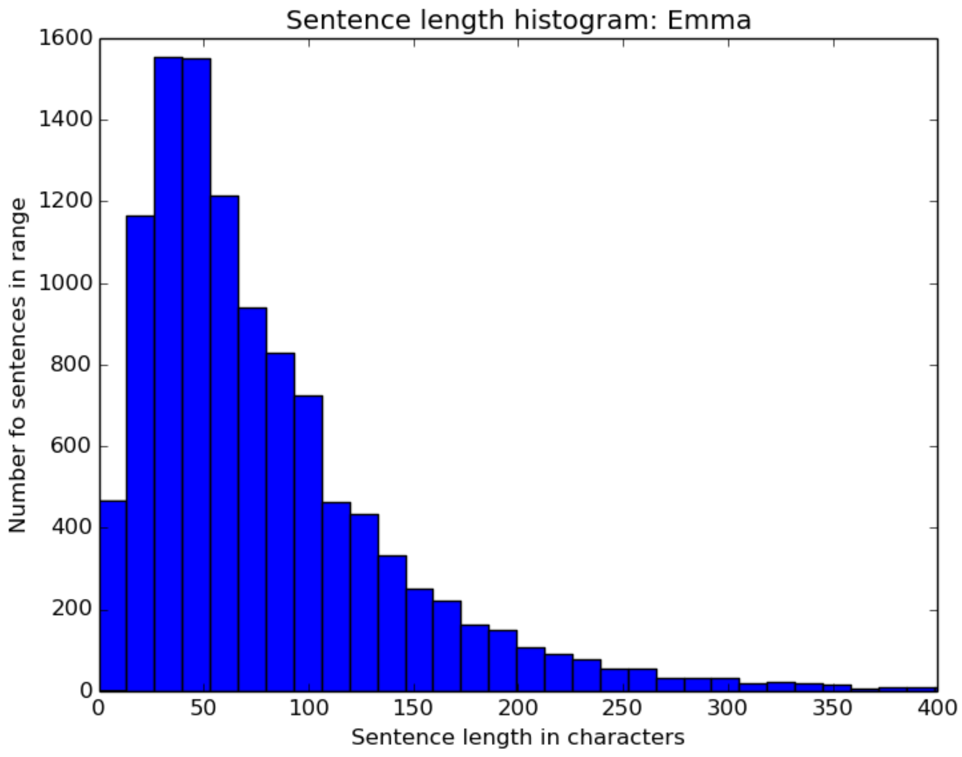
Comparing similarity text file created by my program:

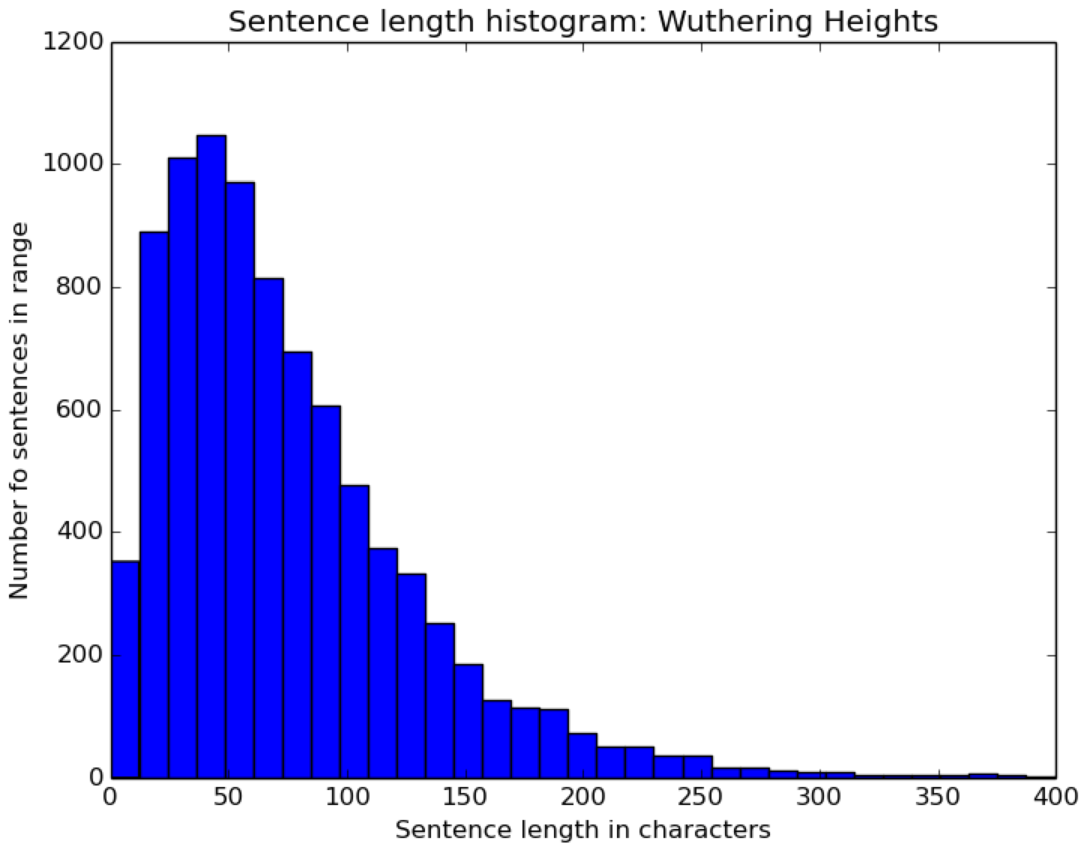
```
judy_november_chopin.txt is 0.8816 similar to Austen-Emma
judy_november_chopin.txt is 0.8882 similar to Austen-Love_and_friendship
judy_november_chopin.txt is 0.8763 similar to Austen-Persuasion
judy_november_chopin.txt is 0.8755 similar to Austen-Sense_and_sensibility
judy_november_chopin.txt is 0.9424 similar to Bronte-Wuthering_heights
judy_november_chopin.txt is 0.9112 similar to Dickens-A_tale_of_two_cities
judy_november_chopin.txt is 0.9386 similar to Dickens-Great_expectations
judy_november_chopin.txt is 1.0000 similar to judy_november_chopin.txt
```

This was a half-completed novel I wrote for Nanowrimo and it is most similar to my recent favorite Wuthering Heights! It surprised me how similar they all are to each other.

Graphs are even more interesting:

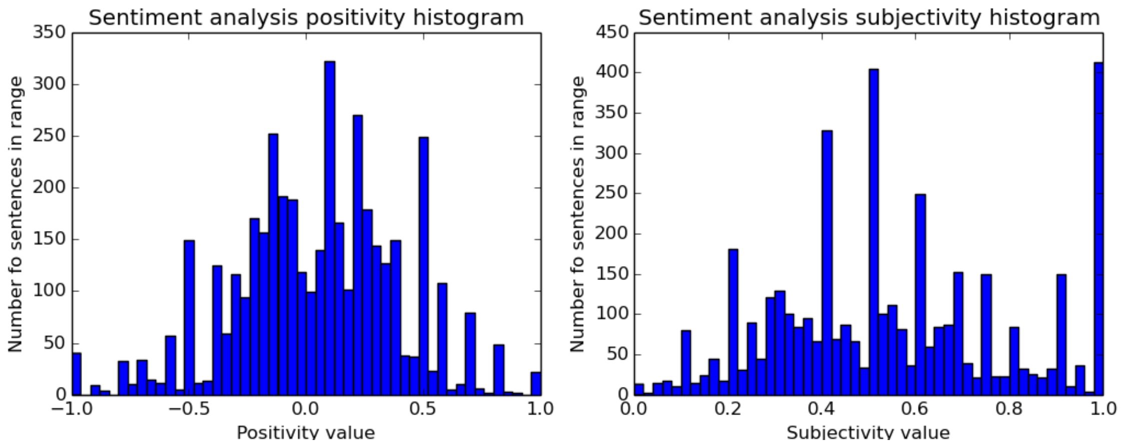




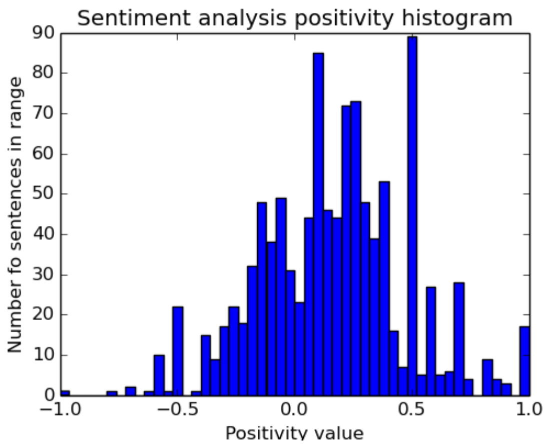


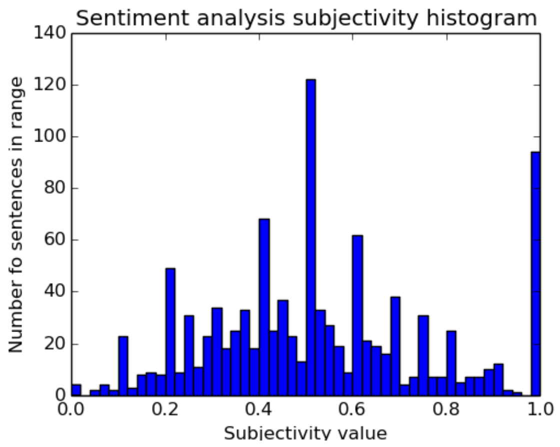
On sentiment analysis:

For Wuthering Heights:



For my novel:





Reflection:

I should have given more thought to what I want to get out of my program before starting to write it, because I wrote some code and didn't use it because that wasn't what I wanted.

It was appropriately scoped because I had individual parts that I can choose to implement depending on my time.

I could have done better with unit testing. Because each function was relatively straightforward, I used print statements and only verified that the output for some examples were good.

If I had more time, I shall fix the Unicode problems with French.