Colvin Chapman
Text Mining Project Writeup

Project Overview:
I used for books in the making of my function found on gutenberg.org. I used the NLTK Package to parse through Merriam Webster's Unabridged Dictionary. I also parsed through three other texts – Oliver Twist, David Copperfield and Sherlock Holmes – to compared letter frequency between them.

Implementation:
To avoid numerous requests for downloading entire books, I decided to save my books on my disk. I pickled the dictionary since this required some special work to separate the words from the word definitions. Luckily, the dictionary was formatted such that the only words that were made of capitalized letters were the words I wanted (not the definitions). This is is stored as a list with each word representing an element of the list.

I analyzed these scripts by first trimming the excess characters like punctuation, numbers and Greek symbols. The resulting strings were then processed using the equivalent of a 'histogram' function that we made in the reading journals and stored into a dictionary. I then took two approaches to analyzing similarities in letter frequency. The first is organizing a string of characters in order of most to least common in the book and running a levenshtein sequence on it. This tells us how far off the order of letter frequencies was between two works. The second way was making a quantitative analysis on how far off the frequencies were for each letter. This incorporated the difference in frequency between the two groups for each letter.

Results:
I found that when comparing qualitatively, the dictionary has an extremely different order of letter frequencies. I backed this up by taking into account the differences quantitatively, the dictionary remains very different than the written works. Also, the variation in word choice between authors cannot be explained solely by variation in letter frequency. The smallest variation was between two different authors of the same time period – not between Dickens' two books Oliver and David.

Letter Order (greatest to least frequent):

Dictionary:	'EAISRTONCLPDMUHBGYFWVKJZXQ'
Oliver Twist:	'ETAOINHRSDLUMWCGFYPBVKXJQZ'
David Copperfield:	'ETAOINSHRDLMUWCYFGPBVKXJQZ'
Sherlock Holmes:	'ETAOINHSRDLUMWCYFGPBVKXJQZ'
The Bible:	'ETHAONSIRDLUMFWCYGBPVKJZXQ'

Works Being Compared:	**Levenshtein Value:**	**Resultant Value:**
Dict vs Oliver :	20	6.919
Oliver vs David :	6	1.087
David vs Dict:	20	6.673
Dict vs Sherlock:	20	6.925
Oliver vs Sherlock:	4	0.943
David vs Sherlock:	4	1.075
Bible vs Sherlock:	12	2.692
Dict vs Bible:	12	8.839

Reflection:
I could improve the format of how my code displays its results. Right now, It's a little bit messy, and not completely intuitive as to what you should put into which function to get a correct output. I think the code itself is very sound as well as my creative take on analyzing text. I gave up on unit testing for things that required exporting and pickling and everything. This was partly because I dumped a ton of time into the pickling toolbox and still couldn't get it running. I'll definitely use some of the string processing tricks I learned moving forward, and I'll make an effort to get to ninja hours to fix up my toolbox so that time wasn't wasted.