Peyton Skwarczynski

CSI 4180 – Natural Language Processing

Dr. Steven Wilson

NLP Homework 0: Counting Tokens

1. Data:

   The input file I selected for natural language normalization was a text-only copy of the novel *Great Expectations*, by author Charles Dickens. I downloaded this file from the website Project Gutenberg (https://www.gutenberg.org/). An interesting topic that was presented from this novel is that since it is from old English literature, there are a lot of hyphenated and contraction words that are no longer used in modern day language. This inclusion caused for an increase in unique tokens, as there is a seemingly endless number of concatenated words you can create by utilizing contractions in old English literature.

2. Methodology:

   The approach I took to creating my script was to initially decide what optional choice of normalization I wanted to include. Since my novel was *Great Expectations*, being from old English literature, I chose to include punctuation removal as my fourth type of text normalization. From there, I initially researched how to use argument parsing to allow the user to customize the types of text normalization when running the script. After I finished coding that first step, I needed to create a format that allowed the user to select any possible combination of arguments. To do this, I utilized if loops to check for the corresponding arguments. Within each prospective loop, I would include the proper code to undergo the specific type of text normalization that the user requested. After those loops, I needed to include code to display the token and their corresponding frequencies in a text file, as well as open a graph to visually display that data. I was able to research and then create code that opens an output text file, repetitively loops to write the tokens and their frequencies to that file, and also generates a graph using matplotlib to display a graphical representation of such data.

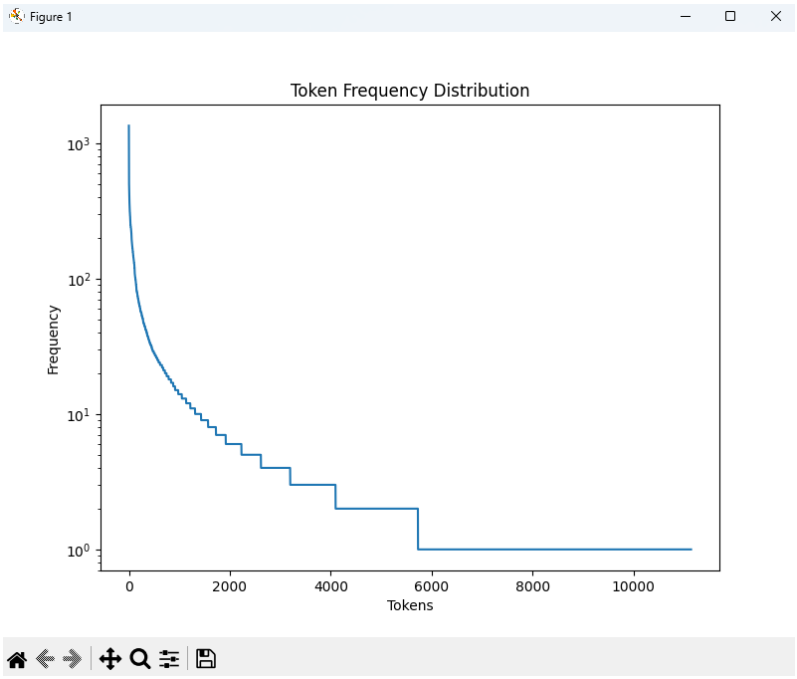
   The specific options I included within my code were lowercasing, lemmatization, stop word removal, and punctuation removal. When running the script, the user has the option to choose any combination of those four options, including not selecting any normalization options at all. The additional option that I chose to add to my program was punctuation removal. This became a useful function because for some reason, the book had interesting spacing that caused the punctuation to be separated from words, ultimately allowing them to be identified by the program as unique tokens. Since there are far fewer punctuation marks than there are words, many forms of punctuation would end up being output as the most frequently occurring tokens in the document. By allowing the user to remove all forms of punctuation, the user is then provided a much cleaner and accurate data set of normalized tokens.

3. Sample Output:

   With the input text file provided in GitHub, the following data was collected when utilizing all optional forms of text normalization in my program.
   Command - py hw-0.py input.txt -pr -sr -le -l

Output –

First 10:		Last 10:	
said	1346	saddened	1
mr	872	belongs	1
joe	684	built	1
would	594	wandereryou	1
one	503	thereforeyes	1
could	476	incompatible	1
time	448	nownow	1
hand	442	buti	1
know	399	hopeinto	1
come	390	tranquil	1



4. Discussion:

One thing I noticed about the words at the top of my list is that half of them are verbs. Instead of the most frequent normalized tokens being common nouns, the majority of them represent actions that took place. As for the most frequent token, “said,” this makes sense to have the highest frequency, as this book contains heavy character dialogue. Additionally, the second and third most common tokens, “mr” and “joe,” represent characters within the book, so these tokens would realistically be repeated numerous times. When looking at the least frequently used normalized tokens, half of them turn out to be words that aren’t actually real. This is due to the old English literature that this novel was written in. Since a lot of the dialogue amongst characters represented the traditional English language, there was a heavy usage of punctuation like hyphens and apostrophes to combine words. During the normalization of these tokens, punctuation was removed, causing these words to be combined.

As for my data's relation to Zipf's law, there is little correlation. In Zipf's law, there seems to be a strong negative linear trend in the relationship between the frequency of the token and its rank. Looking at my printed graph, this is not the case with the data. I believe that the difference between my gathered data and Zipf's law could be due to the fact that my novel contained heavy amounts of dialogue. During normal, everyday speech, people tend to use simple and non-complex language, which would cause a small group of words to have a much higher frequency than others. This could easily explain the large drop-off seen in my displayed graph. Since there tends to be a generally small number of words that are most commonly used within everyday speech, the effect that stop word removal has on token normalization is large. By including this option, the user is able to remove words that carry little meaning, but on the other hand drastically shrink the overall sample size of the most commonly used tokens. In general, regular content words are used less frequently than stop words, so by removing such a large number of words, the user is heavily altering the overall sample of tokens within a given text.

Overall, I was able to learn a lot from completing this assignment. Last semester I did not have any courses that involved programming, so I have been out of practice for a while now. Additionally, I have zero experience programming in Python, so it took a lot longer than expected to complete this assignment. Not only did I have to familiarize myself with the proper syntax, but I also had to touch up on data structures and do research on various python libraries I could utilize within my program. Python syntax is quite different from what I am used to, since I mainly have experience coding in Java, so this assignment contained a large learning curve for myself. It was also difficult learning how to efficiently use all the imported python packages I included within my code. There were certain features I thought to be necessary for this program, but I initially did not know how to complete these tasks. Ultimately, these problems were solved by doing a lot of research.