NLP Homework 0: Counting Tokens

Emma Ozias

1. Data

I selected two different text files. The first file which I named "ebookOne.txt" is *The Great Gatsby.* The second file which I named "ebookTwo.txt" is *The Adventures of Sherlock Holmes.* I downloaded both of these books from Project Gutenberg.

2. Methodology

First, I defined the four different command line arguments that the user could enter (--lower, --removePunct, --lemma, --removeStopWords). I also added code to open and read a file from the command line. The text from the file is then stored in a variable called content. Next, the program begins the preprocessing based on what is entered in the command line. Lowercasing is completed first. All characters in the content variable are lowercased. Next, punctuation is removed. In order to remove the punctuation, I tokenized the variable content, and defined a regular expression to decide whether the punctuation needs to be removed or not. My regular expression does not remove punctuation from contractions, but it removes all other punctuation. After my regular expression finishes removing the punctuation, I join the tokens back together and store the text in the content variable. Next, the text is lemmatized. I used the NLTK library to properly lemmatize my text. Finally, stop words are removed from the text. Uppercase and lowercase stop words are removed. I used the NLTK library's stop word list. After all of the preprocessing has been completed, I count how many times each word is seen in content. Then, a list of the most frequent to least frequent words is printed into the output.txt file. Next, the variable content is split and turned into a set to get the number of unique tokens. Finally, a graph with words on the x-axis and frequency on the y-axis is created.

My additional preprocessing option is --removePunct. I think that this option is useful because when punctuation is not removed, it often shows up at the top of the word frequency list. The frequency of an exclamation mark is not something that I want to be represented on my word frequency graph because it is not relevant. Punctuation is helpful

for text classification, but that is not the focus of this assignment. Therefore, it is best to remove it.

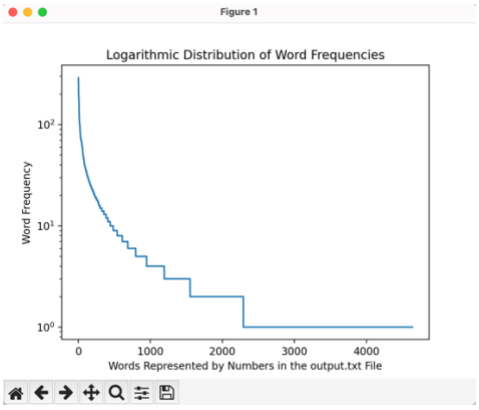
3. Sample Output

Sample output using ebookOne.txt:

```
Terminal  Local
MacBook-Pro:CSI4180_Homework0 emmaozias$ python3 homework0.py ebookOne.txt --lower --removeStopWords --removePunct --lemma
The total number of tokens in this txt file before preprocessing is:
60587

Converting to lowercase...
Removing punctuation...
Lemmatizing the text...
Removing stop words...

The number of unique tokens in this txt file are:
4640
```



homework0.py output.txt

```
1: say: 291
2: gatsby: 264
3: go: 261
4: come: 207
5: tom: 191
6: daisy: 186
7: get: 181
8: look: 179
9: know: 173
10: one: 152
```

homework0.py output.txt

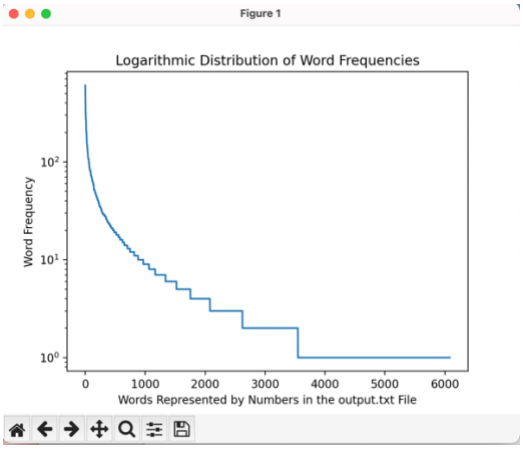
```
4630: pander: 1
4631: greatest: 1
4632: transitory: 1
4633: aesthetic: 1
4634: contemplation: 1
4635: commensurate: 1
4636: obscurity: 1
4637: republic: 1
4638: orgastic: 1
4639: recede: 1
4640: current: 1
```

Sample output using ebookTwo.txt:

```
Terminal  Local
MacBook-Pro:CSI4180_Homework0 emmaozias$ python3 homework0.py ebookTwo.txt --lower --removeStopWords --removePunct --lemma
The total number of tokens in this txt file before preprocessing is:
125125

Converting to lowercase...
Removing punctuation...
Lemmatizing the text...
Removing stop words...

The number of unique tokens in this txt file are:
6083
```



homework0.py output.txt

```
1: say: 605
2: upon: 465
3: holmes: 462
4: one: 376
5: come: 350
6: would: 327
7: see: 325
8: man: 305
9: could: 286
10: know: 275
```

homework0.py output.txt

```
6069: dispatch: 1
6070: relieve: 1
6071: assemble: 1
6072: fever: 1
6073: persevere: 1
6074: seaman: 1
6075: blockade: 1
6076: arguments: 1
6077: locus: 1
6078: standi: 1
6079: survive: 1
6080: solely: 1
6081: mauritius: 1
6082: manifest: 1
6083: walsall: 1
```

4. Discussion

The words at the top of both lists include not only common words, but also character names. The names of the main characters in both of the books made the top 10 most frequent words. For example, “holmes” was seen 462 times and “gatsby” was seen 264 times. The words at the bottom of both lists are obscure. They are not part of everyday language. The bottom of the first output includes the words “pander”,

“transitory”, and “recede”. None of these words are part of everyday vocabulary. In fact, I cannot recall the last time that I heard each of these words. Additionally, “locus” and “standi” are at the bottom of the second list. Locus standi is a Latin phrase. Also, the bottom of the second list includes two places: Walsall, England and Mauritius. Despite this, some common words did appear in the lists of least frequent words. For example, “greatest”, “current”, “relieve”, and “survive” are not uncommon words, but they are all only seen once in either *The Great Gatsby* or *The Adventures of Sherlock Holmes.*

Zipf’s law states that the most common word is seen twice as often as the second most frequent word, and three times as often as the third most common word. This pattern is expected to repeat itself for the fourth word, the fifth word, and so on. The most frequent word lists for *The Great Gatsby* and *The Adventures of Sherlock Holmes* do not follow this pattern. However, another important piece of Zipf’s law is that word frequency plots should have a descending staircase shape. Both of my frequency plots have this descending staircase shape. I think that my plots follow Zipf’s law because my text has been preprocessed. If my text was not preprocessed, then the words swimming and swim would not be counted as the same word. All the different deviations of each word could disrupt the staircase shape of my plot.

Removing stop words will decrease the total number of tokens more than removing verbs and nouns. This is due to the fact that stop words always have a higher frequency count than verbs and nouns. For example, “the” could be seen hundreds of times in one book, but swimming likely will not be seen more than one hundred times.

I learned a lot while completing this project. I improved my Python skills. This is my first project that I have completed with Python. I learned how to use command line arguments. In the past, I have never used a command line argument before. Typically, I would have asked the user if they wanted stop words removed. Then, if they said yes, I would remove the stop words. Then, I would ask if they wanted their text lowercased. Again, I would take their input from the console. Also, I learned how to open, read, and write to files using Python. Finally, I also learned how to use libraries in Python. I downloaded the matplotlib library to make my graph. Also, I used the nltk library for preprocessing my text. In the future, I plan on using both of these libraries again.