ARI_425_HW_1

James Ryan

January 2026

1 Data

I chose "War and Peace" by Tolstoy, a famously long and inscrutable novel, for its length and to see what it would ultimately reduce to. "Alice in Wonderland" by Lewis Carroll, is in the github, mostly because it has the Gutenberg Project license and boilerplate text. I removed this from "War and Peace" so I didn't pollute my results. Interestingly, I only discovered the boilerplate text after I processed the file and the word "email" was in the token list. The Gutenberg Project was the source for both books.

2 Methodology

For this project, I used python as the language, and employed the libraries nltk, pandas, maptlotlib and numpy (the last 2 for figures) to make tokenize_me, a Linux command line tool for generating unique token counts from a given sample, and Tokenize_me.ipynb, a Jupyter notebook that I used as scratch and a testbed for the command line tool, and to generate the figures. The command line tool takes the following arguments:

- `--file`, required along with an input file, which defines the sample text to be analyzed
- `--lower`, optional, which lowercases the text before processing
- `--stop`, optional, which removes stopwords found in the nltk stopword list
- `--stem`, optional, which makes use of nltk's Porter Stemmer to reduce tokens to their root words
- `--no-rares`, along with a number, optional, which removes words occurring less than or equal to the number argument

The rare word removal option was added because after the discussion in the January 21st class I wanted to see the actual effect on a longer corpus. As it turns out, this is pretty useful for removing the long tail in the processed token list resultant from rare words in the text, which really is what made the plots in the next section at all readable.

3 Sample Output

.

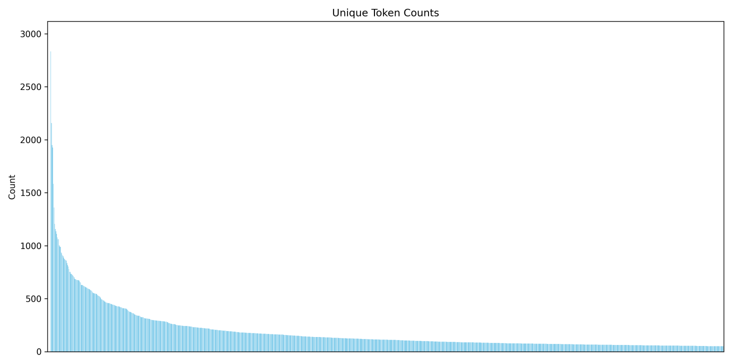

Figure 1: A Readable Plot of the War and Peace Corpus

The following options were used to generate this output: `--lower --stop --stem --no-rares 50`. The reason that 50 was chosen for the rare word count was that it is simply impossible to show useful graphs with the whole corpus usefully in a single plot. Take Figure1 for example.

Table 1: Top 10 Tokens With Counts

Rank	Token	Count
one	2157	
pierr	1949	
princ	1926	
look	1584	
would	1360	
natásha	1205	
man	1157	
andrew	1136	
could	1111	
face	1078	

It is not the whole graph. It really looks more like Figure2. This considered, presenting approximately 10,000 unique tokens (as found in "War and Peace") is simply not feasible in any meaningful way.

Figure 2: Entire War and Peace Corpus Without 50 Rare Words

Table 2: Bottom 10 Tokens With Counts

Token	Count
broke	51
conceal	51
hippolyt	51
insist	51
journey	51
mason	51
mon	51
mysteri	51
offici	51
shook	51

4 Discussion

.

4.1 Assignment Questions

The most common words in the corpus are pretty much as expected, with common words like "one", "look", and "would", as well as the names of the main characters, all appearing in the top 10. "One" is a little interesting as the top choice. I suspect this is an artifact of 19th century speech patterns over modern ones but would need to do more research. Verbs such as "look" and "would" are unsurprising. I suspect if I had not removed stopwords the results would have been even more striking, with, I'm sure, "the" and "a" at the top of the list. The bottom end words don't really share any commonalities but this is to be expected of words that don't appear often.

As compared to the Wiki article, the corpus does not strictly align to Zipf's law, with the top word appearing twice as often as the next word, etc., but (I think) roughly follows the expected form. If I get some time I'll graph this and find out. Again this is likely due to stopword removal. Speaking of stopwords, removal not only significantly reduces the number of word types (with a corresponding but small reduction in the number of tokens), it makes the resulting corpus much more semantically significant.

4.2 Learning

I have done variants on this project before, or at least NLP preprocessing. The rare words concept was new to me, but none of the core python was particularly new. I did spend some cycles on making a command line app, which was fun. "War and Peace" was an easy choice, as noted for its length and inscrutability, but the results were really interesting: Word count on the text file was 3,359,652 individual words, yet this reduced to only 1046 unique tokens after lowercasing, stemming, stopword removal, and removal of rare words (count 50 or less). I wonder whether you could get the gist of the book by reading the whole token list (which I have not done, but might).

I do have to give a shout-out to ChatGPT, which is (for me) indispensable for syntax repair and debugging. My favorite prompt is ”fix this,” which saves loads of time looking up syntax particulars and finding stupid mistakes (or even complex ones).

In case you need it, the GitHub can be found at https://github.com/jjryan111/ARI425