SFWRENG 4NL3 Assignment 1

Sumanya Gulati

20 January 2025

Contents

List of Tables

List of Figures

1 Data

`myfile.txt` is a text file that contains inaugral speeches by United States Presidents' from George Washington to Barack Obama. The dataset was acquired from the Project Gutenberg collection and was downloaded as a Plain Text UTF-8 encoded file.

Prior to tokenization, the dataset was manually pre-processed which included removing the title page, table of contents, disclaimers and such information from the beginning and the ending of the text file to ensure consistency and remove any characters unrelated to the actual content.

I chose this dataset in light of today's inaugral speech of the 47th U.S. President. I also wanted to be able to use the script from this assignment to gather insights about how inaugral addresses have changed over centuries and analyze whether the increased use of words like 'immigration' and 'economy' is a recent trend over the past few decades.

I also want to analyze if the use of words propogating sentiments of nationalism and unity increases in years following major national of global events like the Vietnam war, 9/11, the 2008 recession, COVID-19 pandemic and more. If you would be interested in viewing the results of these analyses, feel free to shoot me an email and I would be glad to share them (assuming, I am actually able to get it done soon).

2 Methodology

This section outlines the approach employed in writing this script and explains available normalization options while describing their use and a rationale for using them.

2.1 Approach

The approach I used to write this script involved breaking it down into steps and implementing one required step at a time, tetsing it and then moving onto the next step. I started by writing a skeletal script that reads a file and implements command-line argument parsing. After that, I wrote the `tokenize_text()` function to pattern-match the input text file and break it into tokens, storing the result as a list.

The next step involved implementing the *lowercase* and *remove_stopwords* options, followed by implementing the *lemmatization* and *stemming* options, the code snippets for which were copied from the lecture slides. The `visualize()` function was then implemented to produce first, a DataFrame (basically, a nicer table to view the output) and then a bar plot. A trial and error approach was used to figure out if either or both axes should be set as a log scale and a log-log line plot was generated to compare my result with that of datasets obeying Zipf's law as shown in this Wikipedia article.

Generative AI was used to resolve a logical bug and write line 12 of the script. A search query weas employed using ChatGPT with the following prompt: *"I am trying to tokenize an input text file using regex pattern matching but some of my tokens in the list are preceeded by the letters, ufeff. Could this be random or is there a logical error in my code?"*. As outlined in the course outline, the carbon footprint of this query is 4.32g of CO_2.

2.2 Available Options

The following options, in any combination, can be used by the user during the text normalization steps:

1. `--lowercase`: Change letter case
2. `--stem`: Apply stemming
3. `--lemmatize`: Apply lemmatization

4. `--remove-stopwords`: Remove commonly used stopwords in the English language

5. `--remove-numbers`: Remove numbers

Beyond the defined requirements, I added the option to remove numbers. Since the text file contains speeches spanning centuries, it seemed like a fair assumption (backed by a brief look at the raw data) that the text file contains the use of digits to write out for example, the date or the year of the speech. I believe this is a useful option as it removes the numbers added as a part of the description, producing a more consistent dataset that focuses purely on the content of the speeches by exluding the remaining descriptive text.

I recognize that this option may cause unwanted issues by removing numbers that are a part of the actual speech or not remove numbers that are spelled out as opposed to being written ad digits. Configuring this option to not cause these issues seems like a more complex task that would require additional time and effort.

3 Sample Output

Using the `pandas` library and the DataFrame structure, a table has been created to sort the list of unique tokens in decreasing order along with their frequency/count.

Figures 1 and 2 display tables showing the top 25 words and their frequency based on the program's output. Figure 1 has been generated using all options except `--stem` and figure 2 has been generated using only the `--lowercase` and `--lemmatize` options.

	Tokens	Count	Rank
0	government	645	1
1	people	606	2
2	nation	484	3
3	u	477	4
4	state	441	5
5	upon	373	6
6	must	363	7
7	may	338	8
8	great	334	9
9	power	334	10
10	country	330	11
11	shall	314	12
12	world	310	13
13	citizen	299	14
14	every	292	15
15	law	268	16
16	time	267	17
17	one	257	18
18	peace	255	19
19	right	252	20
20	new	246	21
21	public	225	22
22	american	223	23
23	would	211	24
24	constitution	211	25

Figure 1: Top 25 words with their count.

	Tokens	Count	Rank
0	the	10030	1
1	of	7064	2
2	and	5246	3
3	to	4507	4
4	a	3183	5
5	in	2778	6
6	our	2138	7
7	it	1949	8
8	that	1785	9
9	we	1697	10
10	be	1473	11
11	is	1447	12
12	for	1187	13
13	by	1081	14
14	have	1007	15
15	which	1003	16
16	with	950	17
17	not	948	18
18	will	874	19
19	i	834	20
20	this	825	21
21	all	807	22
22	are	800	23
23	their	745	24
24	government	645	25

Figure 2: Top 25 (including stopwords) words with their count.

Figures 3 and 4 display tables showing the last 25 words and their frequency based on the program's output. Figure 3 has been generated using all options except `--stem` and figure 4 has been generated using only the `--lowercase` and `--lemmatize` options.

	Tokens	Count	Rank
7948	durably	1	7949
7949	marginalized	1	7950
7950	describes	1	7951
7951	seneca	1	7952
7952	selma	1	7953
7953	stonewall	1	7954
7954	sung	1	7955
7955	unsung	1	7956
7956	footprint	1	7957
7957	preacher	1	7958
7958	inextricably	1	7959
7959	daughter	1	7960
7960	gay	1	7961
7961	student	1	7962
7962	workforce	1	7963
7963	expelled	1	7964
7964	detroit	1	7965
7965	appalachia	1	7966
7966	lane	1	7967
7967	newtown	1	7968
7968	contour	1	7969
7969	absolutism	1	7970
7970	philadelphia	1	7971
7971	realizes	1	7972
7972	p	1	7973

Figure 3: Last 25 words with their count.

	Tokens	Count	Rank
8199	selma	1	8200
8200	stonewall	1	8201
8201	sung	1	8202
8202	unsung	1	8203
8203	footprint	1	8204
8204	preacher	1	8205
8205	inextricably	1	8206
8206	daughter	1	8207
8207	gay	1	8208
8208	student	1	8209
8209	workforce	1	8210
8210	expelled	1	8211
8211	detroit	1	8212
8212	appalachia	1	8213
8213	lane	1	8214
8214	newtown	1	8215
8215	contour	1	8216
8216	absolutism	1	8217
8217	40	1	8218
8218	400	1	8219
8219	philadelphia	1	8220
8220	realizes	1	8221
8221	12	1	8222
8222	10	1	8223
8223	p	1	8224

Figure 4: Last 25 words (inclduing stopwords) with their count.

Using the `matplotlib.pyploy` library, a bar graph depicted in figure 5 shows the top 40 words. It must be noted that the y-axis has been set to a log scale to better understand the generated output. Although a normal bar plot without a log scale did produce a readable graph, the difference between frequencies was not as apparent. This graph was generated using all options except `--stem`.

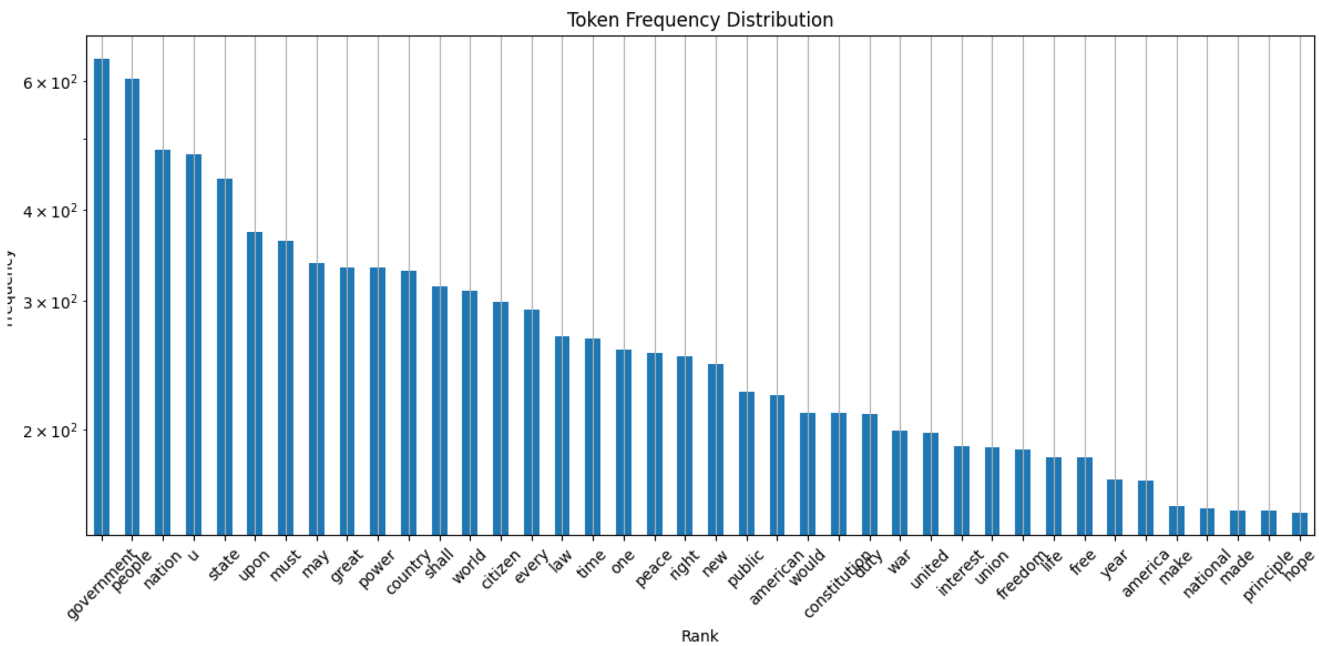

Figure 5: Bar plot showing the frequencies of top 45 words.

To better compare the output with Zipf's Law, a line graph as shown in figure 6 has been generated. It must be noted that this is a log-log plot. This graph was generated using all options except `--stem`.

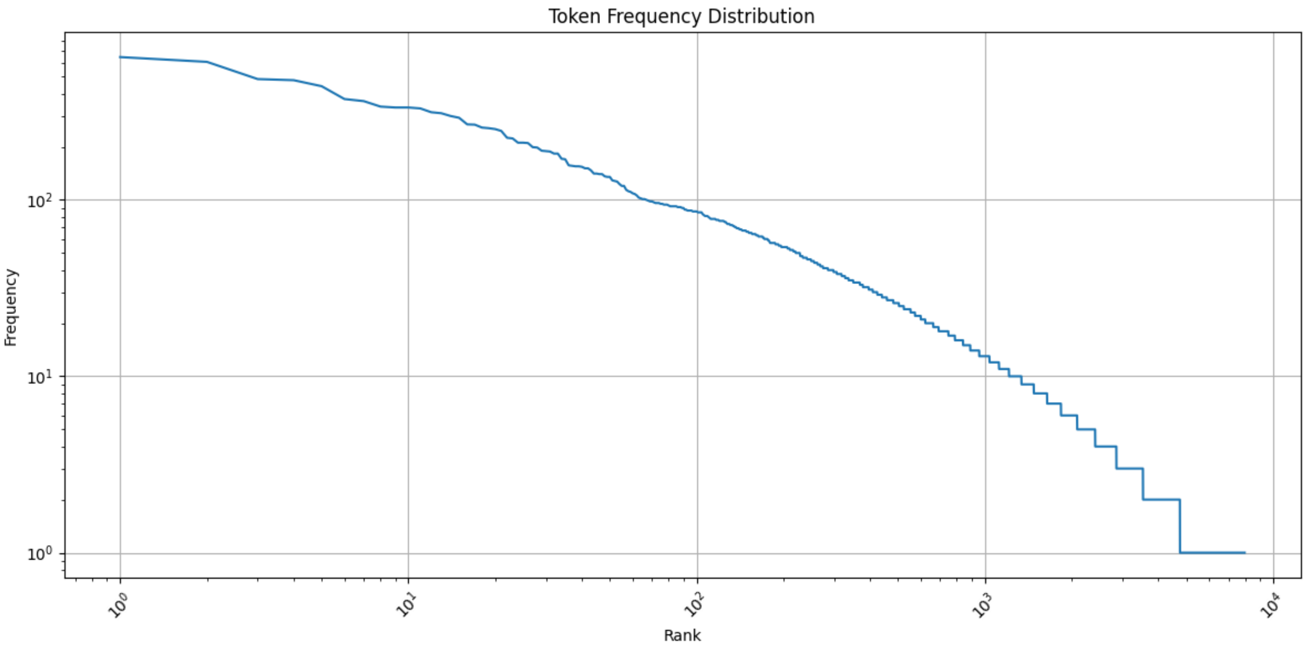

Figure 6: Line plot showing the words and their frequencies.

4 Discussion

This section outlines common findings and insights dervided from the program.

4.1 Findings

On comparsion of the two bar graphs shown in figures 5 and 7, it can be noticed that prior to removing stopwords, all top 40 words consist of stopwords. Once this normalization step has been implemented, however, words truly depicting the contents of the speech start populating the top 40 ranks. The other end of the list, in both the figures 3 and 4 show similar words but with different ranks as the list progresses.

To better compare my reuslts with those mentioned in the article, I generated line plots shown in figures 6 and 8. Figure 8 is very similar to the plots shown in the article and the similarity can be explained by the fact that this list inclduing stopwords complies with Zipf's Law. Figure 6 however looks somewhat different as all stopwords have been reduced thereby removing the compliance with Zipf's Law.

As mentioned earlier, this means that removing stopwords significantly reduces the total number of tokens because they tend to occur frequently. In contrast, removing content words has a smaller impact on token counts but affects the semantic richness of the text more, as content words carry meaning.

4.2 Reflection

5 Appendix

This section icnludes extra figures that might be relevant to the report but have not been referenced in the content.

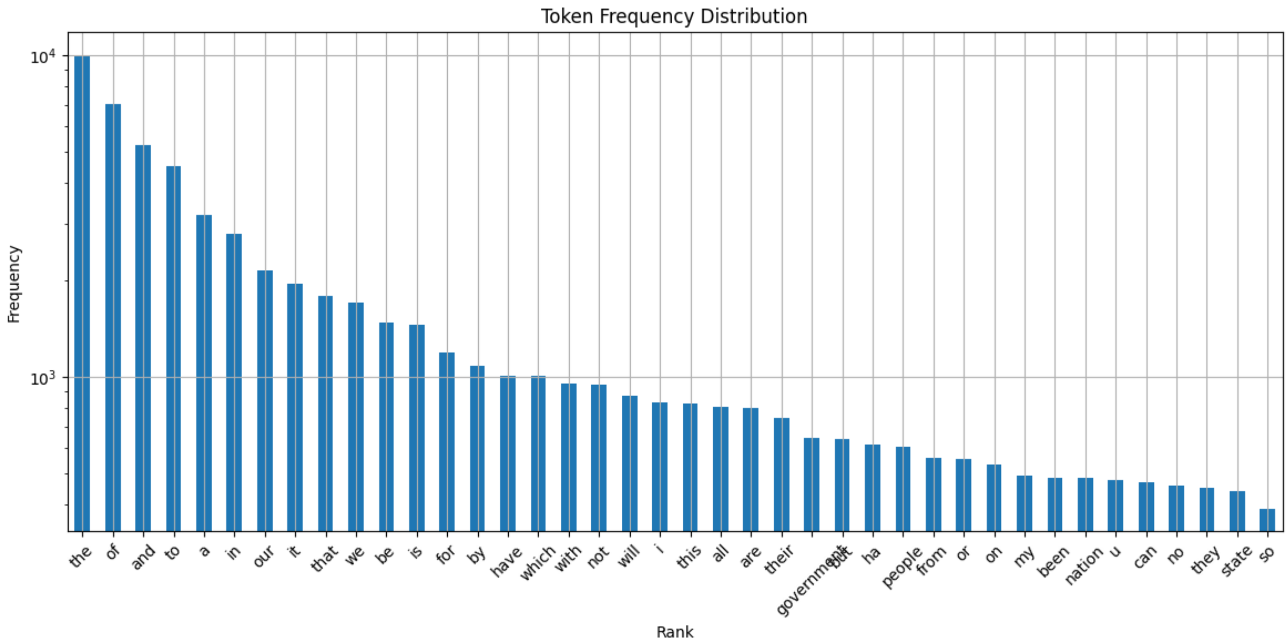

Figure 7: Bar plot showing the frequencies of top 40 words (including stopwords).

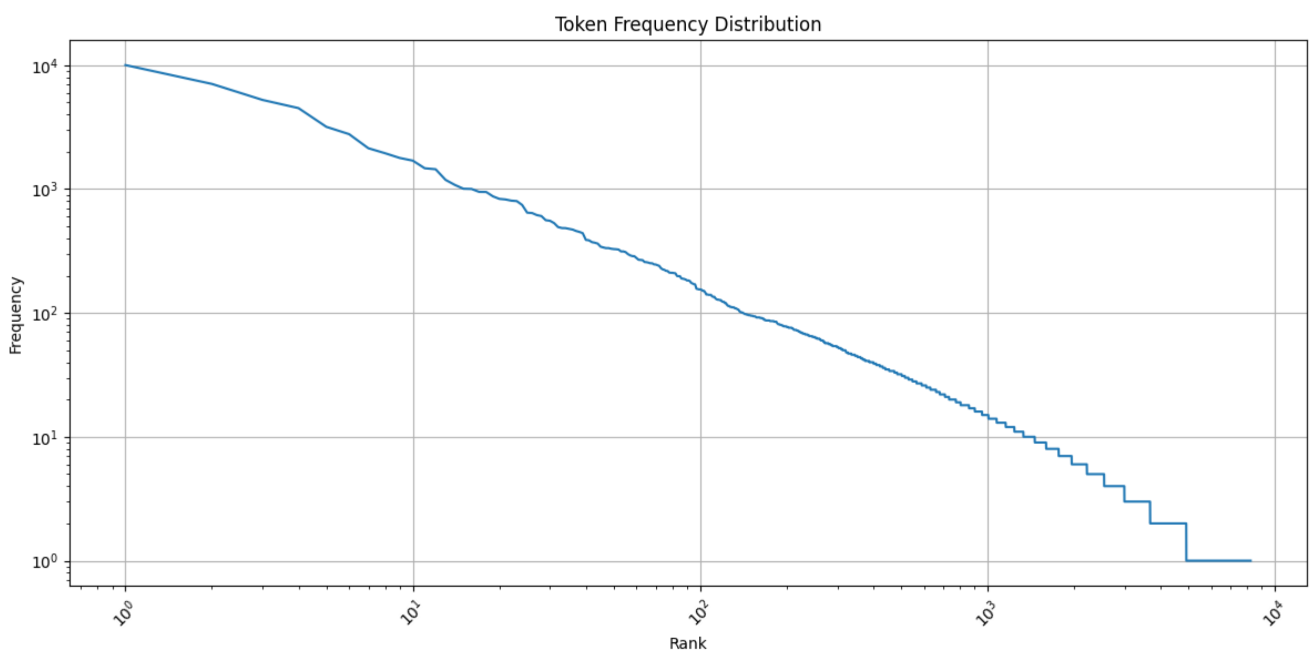

Figure 8: Line plot showing the words (including stopwords) and their frequencies.