Virendra Jethra (jethrav)

4NL3 Homework 1 Report

1. Data:

   The file selected is a combination of the following ebooks:

   i. “The Brothers Karamazov” by Fyodor Dostoyevsky
   ii. “The Wonders of the World” by John S. C. Abbott
   iii. “My Life — Volume 1” by Richard Wagner
   iv. “Twenty Years After” by Alexandre Dumas & Auguste Maquet
   v. “The Boy Mechanic, Book 2: 1000 Things for Boys to Do” by H. H. Windsor
   vi. “Garibaldi, Vol. 1 (of 2)” by Giuseppe Guerzoni

   It contains 55,930 words before normalization in a plaintext version with utf-8 encoding. All the e-books are downloaded through Project Gutenberg. An interesting thing I found about the dataset is that it has a diverse vocabulary from literature, history, and instructions, showcasing various linguistic trends.

2. Methodology:

   a. To start off, I installed the nltk library with ‘pip install’ and imported the necessary libraries such as Counter, re, argparse and pyplot. Next, I installed the NLTK data required for the stemming and lemmatization.

      I made a function for preprocessing text which took in the file path and the arguments for normalization. The function reads a text file, tokenizes the words, applies the specified preprocessing steps, and generates token frequency counts. Additionally, for my customization, I made an option to filter by the minimum length of the tokens.

      For visualization, I made a function that took in the sorted tokens list and displayed a bar graph with the top 30 tokens, with ranks on the x-axis, labelled as the token itself and frequencies on the y-axis. In the main function, I used the argparse to get the user input, print the sorted token list and print the pyplot for the token visualization.

   b. These are the following options the user can perform with a text file:

      i. Lowercasing: This allows the users to lowercase the tokens to have no case sensitivity for consistent token analysis and frequency counting.
      ii. Stemming: This allows the user to reduce words to their root form like "running" to "run" and group similar word variations together for frequency counting.

iii. Lemmatizing: This allows the user to transform words into their dictionary form, like "better" to "good" which preserves the meaning and groups them together for accurate frequency counting.

iv. Removal of Stopwords: This allows the users to filter common, non-informative words like "the" and "is," to focus on meaningful tokens.

v. Minimum Length of Tokens: This allows the users to remove short, irrelevant tokens like "a" or "la" to ensure the analysis emphasizes on longer, more meaningful tokens.

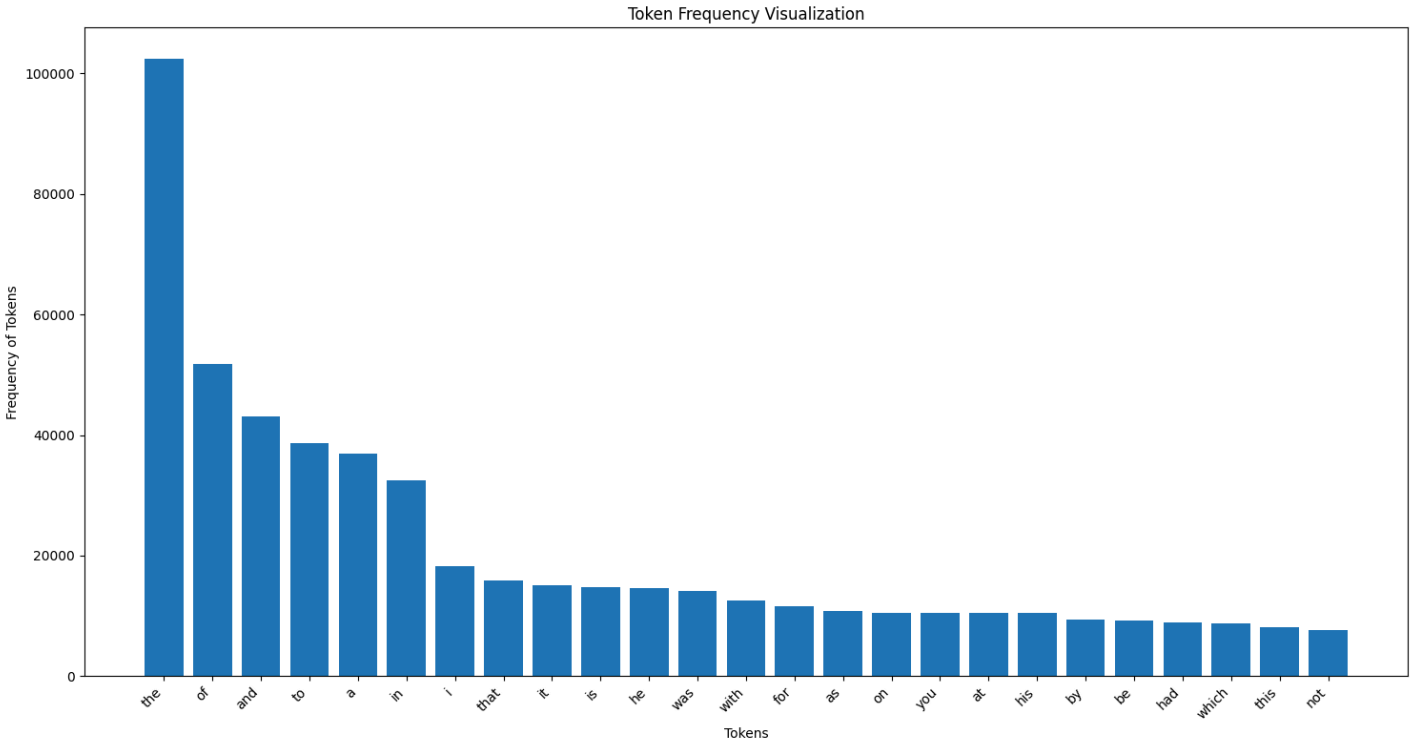
c. The additional option I have added is the “Minimum Length of Tokens.” This is useful as it helps focus on more meaningful words that have not been removed through stemming, lemmatizing, or stopword removal. When I applied all the options, I was still getting words like “e,” “di,” “la,” and “il” as some of the highest-ranking words. However, with the minimum length option, the user can restrict it so that only words with a length of 3 or more appear in the visualization, making the analysis clearer and more focused for the user.

3. Sample Output:

a. The following is the output for the largetokens.txt with the lowercase option for the first 25 words:

the: 102478
of: 51853
and: 43124
to: 38664
a: 36837
in: 32432
i: 18302
that: 15882
it: 15044
is: 14697
he: 14571
was: 14106
with: 12591
for: 11602
as: 10880
on: 10541
you: 10497
at: 10473

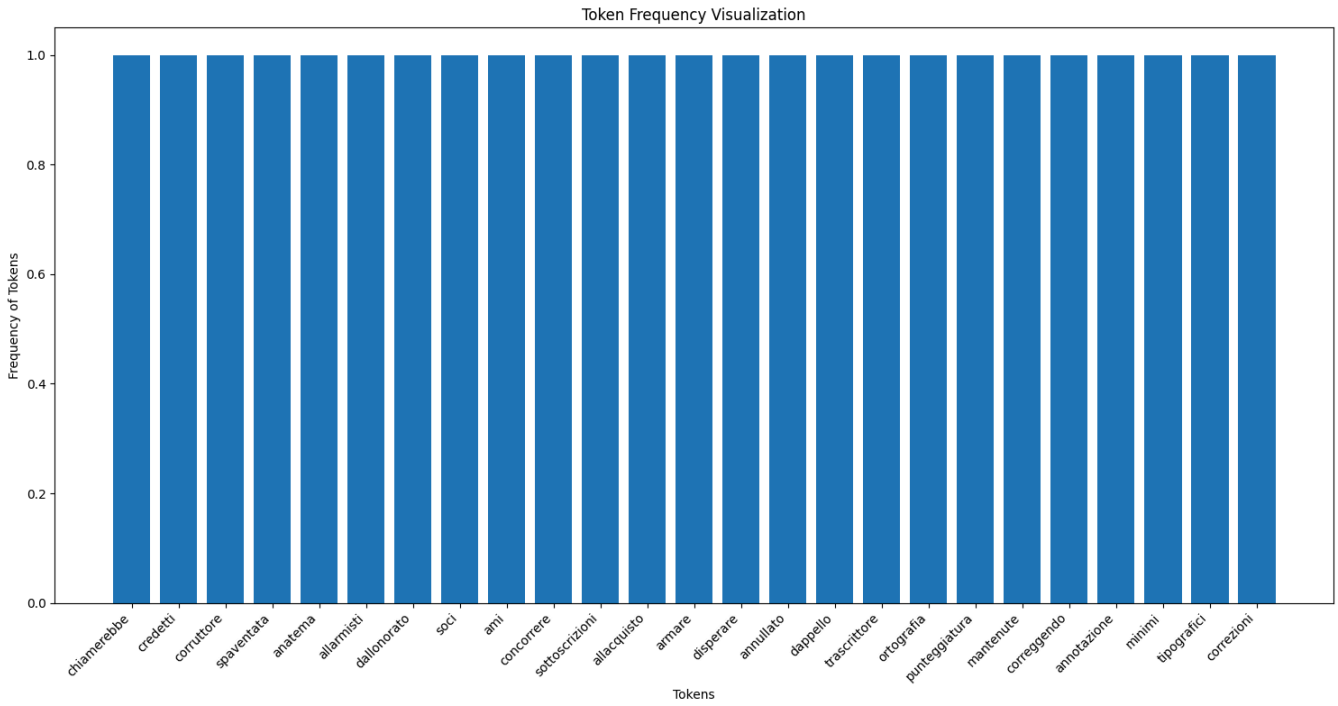
his: 10459
by: 9354
be: 9187
had: 8943
which: 8758
this: 8180
not: 7718



The following is the output for the largetokens.txt with the lowercase option for the last 25 words:

chiamerebbe: 1
credetti: 1
corruttore: 1
spaventata: 1
anatema: 1
allarmisti: 1
dallonorato: 1
soci: 1
ami: 1
concorrere: 1
sottoscrizioni: 1

allacquisto: 1
armare: 1
disperare: 1
annullato: 1
dappello: 1
trascrittore: 1
ortografia: 1
punteggiatura: 1
mantenute: 1
correggendo: 1
annotazione: 1
minimi: 1
tipografici: 1
correzioni: 1



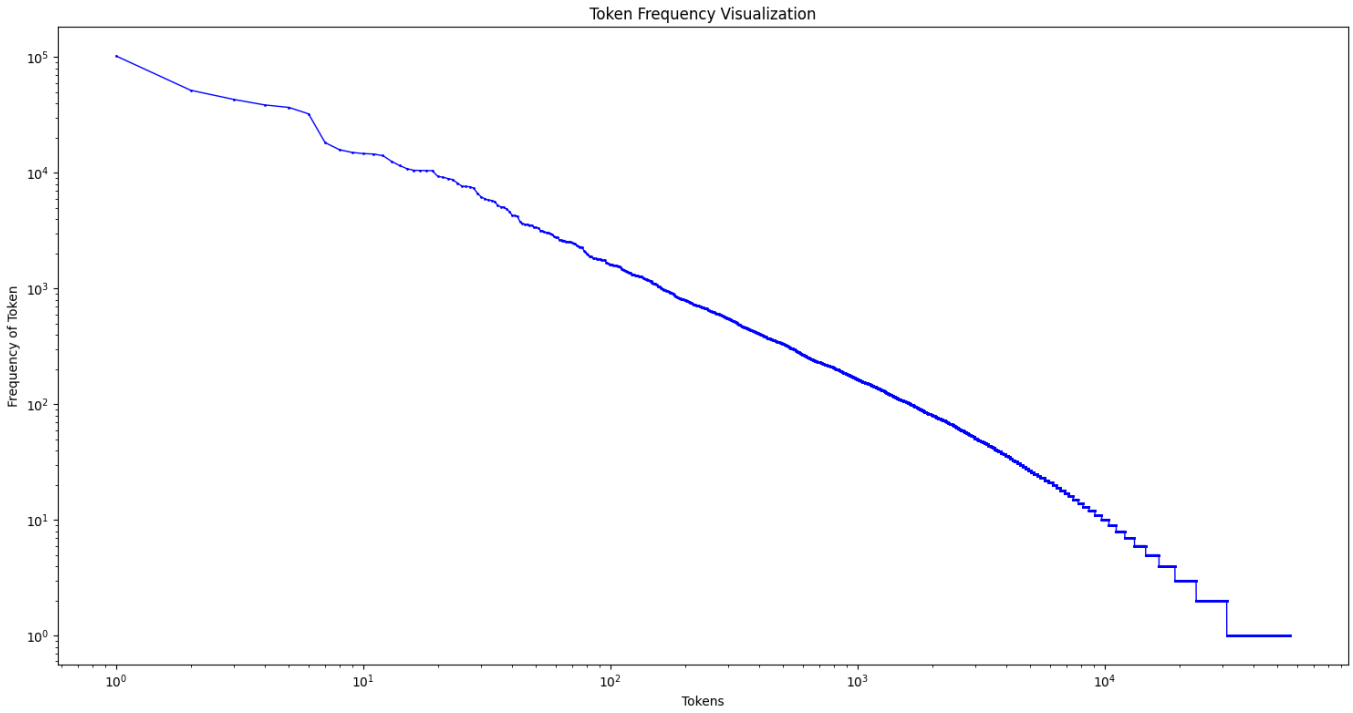
**For figures using stemming, lemmatizing, removal of stopwords and the minimum length of tokens. See the Figures subdirectory.

4. Discussion:
    a. The top words are short and functional words, that are serving as articles, prepositions, conjunctions, pronouns, and verbs in the given text, making them appear frequently. At the end of the list, the words are complex, with

specialized meanings. These words are specific to the given text and reflect niche topics and also can be non-English words.

As you can see in the figure below my results closely follow the Zipf distribution. The curve shows a steep decline for high-frequency tokens and a staircase-like pattern for low-frequency tokens, aligning approximately with the expected rank-frequency distribution. However, there could be differences as the corpus depends on various factors like the text's topic, author style, or preprocessing steps like removing stopwords.

Stopwords like “the” significantly affect the total number of high-frequency tokens in the corpus so removing it leads to a steeper drop in the plot’s initial slope. On the other hand, removing content words affects mid-to-low-frequency tokens, redistributing word counts and altering the plot’s middle and tail regions. Since stopwords dominate the high-frequency end, their removal has a more noticeable impact on the total token count.



b. Throughout this assignment, I gained significant experience with the NLTK library, understanding how to extract meaningful information from a given text using various tools it provides. While setting up and running the library was straightforward, I experienced a few challenges, particularly when removing punctuation and specific characters such as '-' and '/'. To tackle

this, I researched and designed a regular expression that removes punctuation and only selects alphabetical characters. Additionally, I learned how to utilize the argparse module to get user input from the command line, which was required for dynamic execution. Furthermore, I used AI assistance to sort the list of tokens in descending order and print the results. Overall, this experience helped me better comprehend text processing by introducing me to practical methods and resources for working with text data.

Generative AI:

I used the ChatGPT model to ask how to sort the token list with two variables. The carbon footprint is 5.00g of CO2 (model: ChatGPT, Hardware: GTX 1660 Ti, Time Used: 0.5h, Provider: Google Cloud Platform, Region of Compute: Us-east1)