Natural Language Processing

CSE 3201

Project Report by

Team: Ctrl Alt Elite

Nirnay Korde(20UCS133)

Raghav Khanna(20UCS153)

Sourav Jain(20UCS198)

Prateek Jain(20UCS145)

Course Instructor

Dr. Sakthi Balan Muthiah

Associate Professor, Dept. of CSE

Department of Computer Science Engineering

The LNM Institute of Information Technology, Jaipur

Abstract

Natural Language Processing is a subfield of artificial intelligence that deals with interactions between computers and human users. It is basically to find a way to make the computers process and analyze the natural language data.

This project aims to use the existing libraries of Python, to know the various algorithms used behind the processing of text, and also to get a clear understanding of how the algorithms are used in real life.
This report has a summary of the various steps included during the code part of our research, along with visualizations by graphs for easy inferences for a better understanding of the text as well.

LNMIIT
The LNM Institute of Information Technology

Table of Contents

LNMIIT
The LNM Institute of Information Technology

Problem Statement

This project involves various questions to be solved, which centers around various NLP topics such as,

a) Clean the Data
b) Tokenization of text
c) Creation of Word Cloud
d) Giving appropriate tags to the word, PoS Tagging

Data Visualization is also a key part of this project, which we use to get a clear understanding of the text so that we can accordingly prepare for the text processing steps. It is also used to get various inferences of the data after processing for an easy understanding.

All of the Language Processing in this project is done using the **Python** programming language, which has a variety of libraries available for the same. We'll be using the Natural Language Toolkit, commonly known as the **NLTK library** for the text processing, along with data visualization libraries such as **matplotlib.pyplot, WordCloud.**

Data Description

The text used for this project is a course book downloaded in text format of the Operating System Concepts 9th Edition : Abraham Silberschatz . We've used this book for the purpose.

Book Description –
No. of Characters: 17,20,614
No. of Words: 2,99,138

LNMIIT
The LNM Institute of Information Technology

Text Pre-Processing Steps

Text Pre-processing is an important part of Natural Language Processing tasks. It helps to make the data in a more digestible form, which can thus be easily worked upon by the machine learning algorithms.
We used the Python libraries

- **'nltk' for text processing**
- **'matplotlib' for visualization.**

The tasks in Pre-processing we've performed in our code are:

a) Removing content irrelevent to overall book's theme.
b) The following sections has been removed- copyright, preface, index, credits.
c) **Removing Punctuations and Normalization**:
    - We firstly removed all the punctuation characters such as " **!, (,), ;** " etc characters from the entire text novel.
    - Next, we normalized the text.
    - This means we convert the text in a standard form which becomes easy to process.
    - We turned the entire text into lower case letters.

- Converting text into clean-text and removing irrelevant text like figure numbers, roman numbers, links etc. with the help of Regular Expressions(RE).

```
import re

#Creating a string which has all the punctuations to be removed
punctuations = '''!()-[]{};:'"\,<>./?@#$%^&*_~'''
cleantext = ""
for char in text:
    if char not in punctuations:
        cleantext = cleantext + char

#Converting the text into lower case
cleantext = cleantext.lower()
cleantext = re.sub(r'[ ]?this page intentionally left blank[ ]?','',cleantext)
cleantext = re.sub(r'chapter[ ]?[0-9]+','',cleantext)
cleantext = re.sub(r'figure[ ]?[0-9]+','',cleantext)
cleantext = re.sub(r'preface','',cleantext)
cleantext = re.sub(r'^m{0,3}(cm|cd|d?c{0,3})(xc|xl|l?x{0,3})(ix|iv|v?i{0,3})$','',cleantext)
cleantext = re.sub(r'[ ]e[ ]','',cleantext)
cleantext = re.sub(r'[0-9]+','',cleantext)
cleantext = re.sub(r' www[a-z]+ ','',cleantext)
cleantext = re.sub(r' [^aci] ','',cleantext)
```

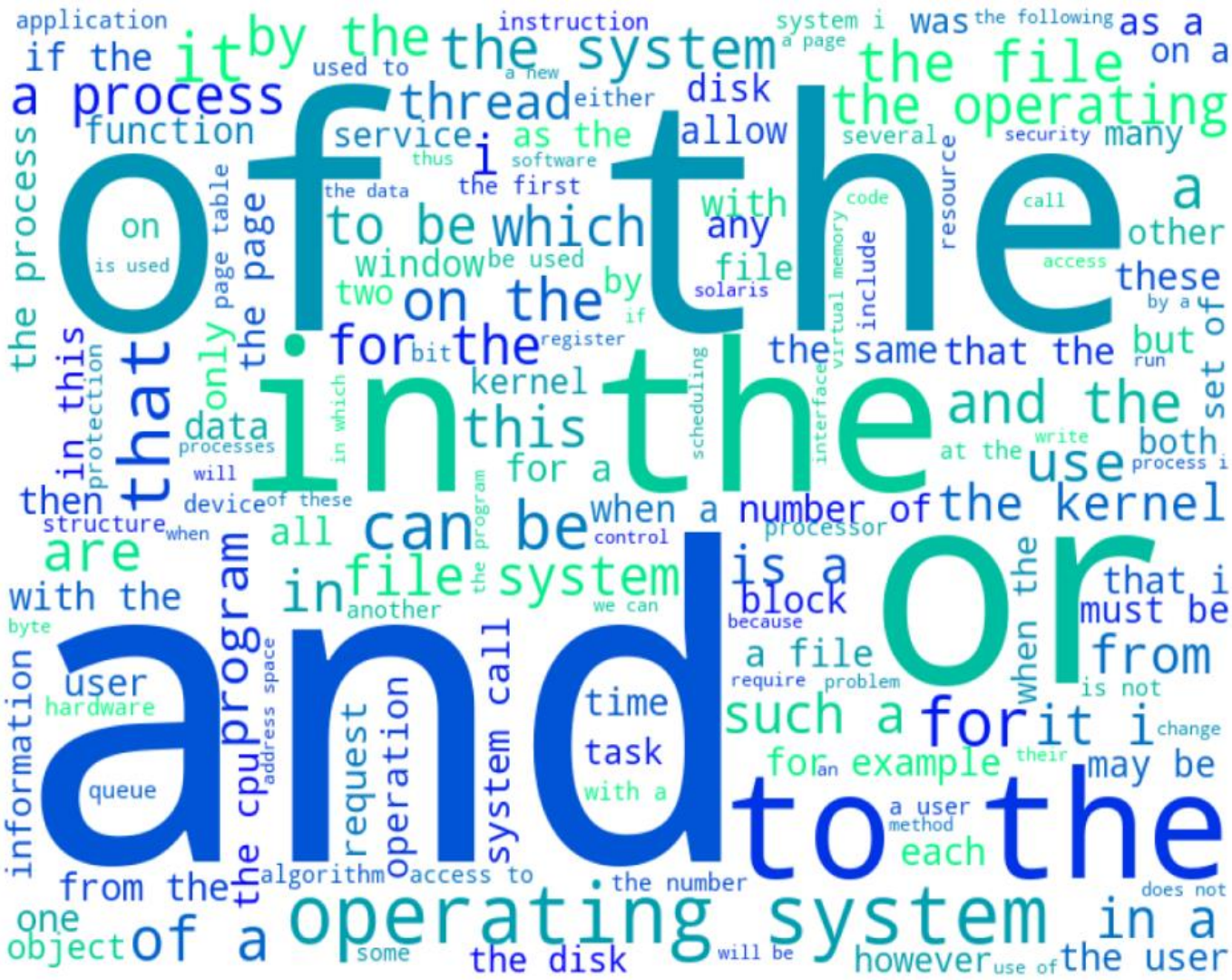
- Length of cleantext : 16,60,217

➢ We can see that the difference between text and clean text is 60,397. This much length of text was occupied by the text which is not of our interest in this project.

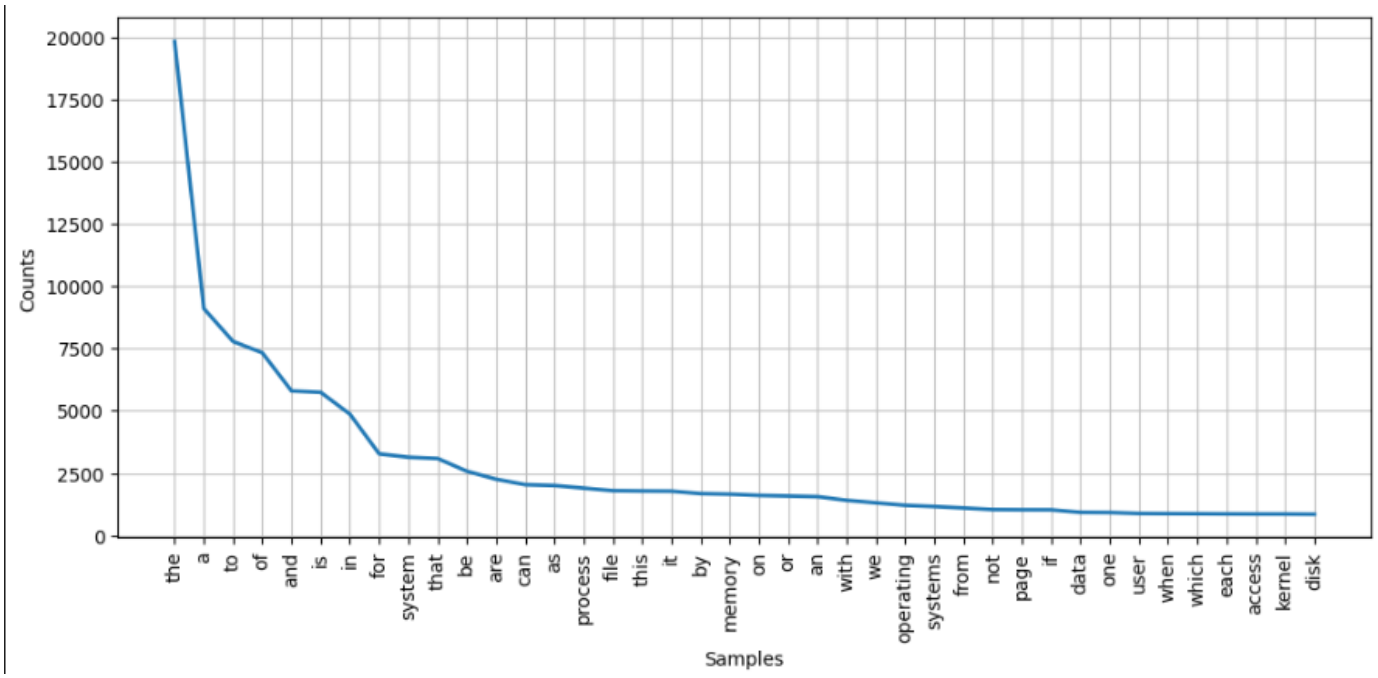
d) Next, we tokenize the cleaned text using the function **'word_tokenize'** from the **nltk.tokenize library.**

- Tokenizers divide strings into lists of substrings.
- This particular tokenizer 'word_tokenize' requires the **Punkt sentence tokenization model** to be installed.
- This Punkt sentence tokenizer divides text into a list of sentences by using an unsupervised algorithm to build a model for words.

- **WordCloud of Cleantext without removing stopwords.**



- **Frequency distribution of cleantext without removing stopwords**



➢ **INFERENCE:**

- As we expected that the cleantext without removing the stopwords will have high frequency of words like **‘the’,’a’,’to’** etc. which can be seen through wordcloud and frequency distribution graph.

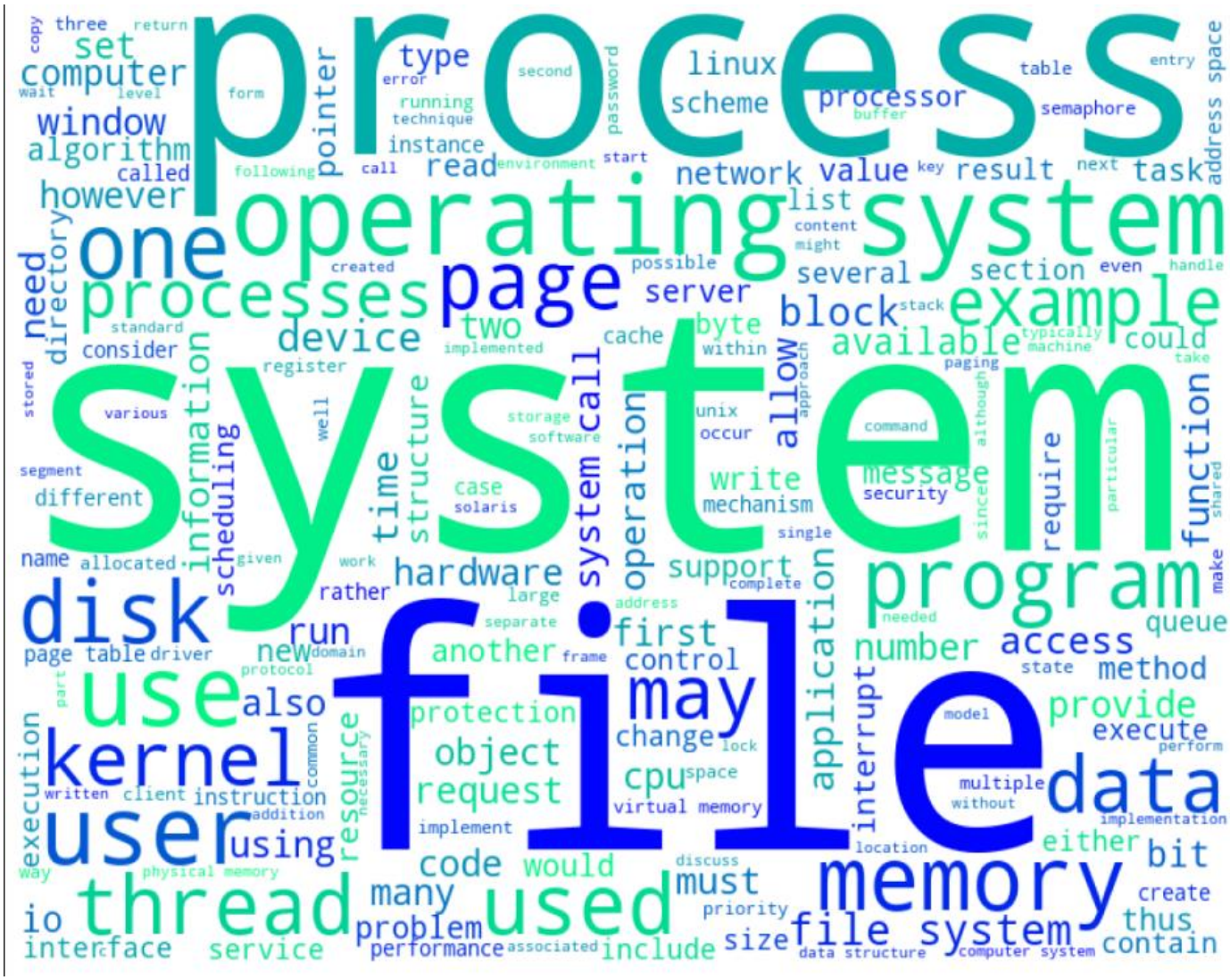
LNMIIT
The LNM Institute of Information Technology

e) Removing Stopwords from Cleantext

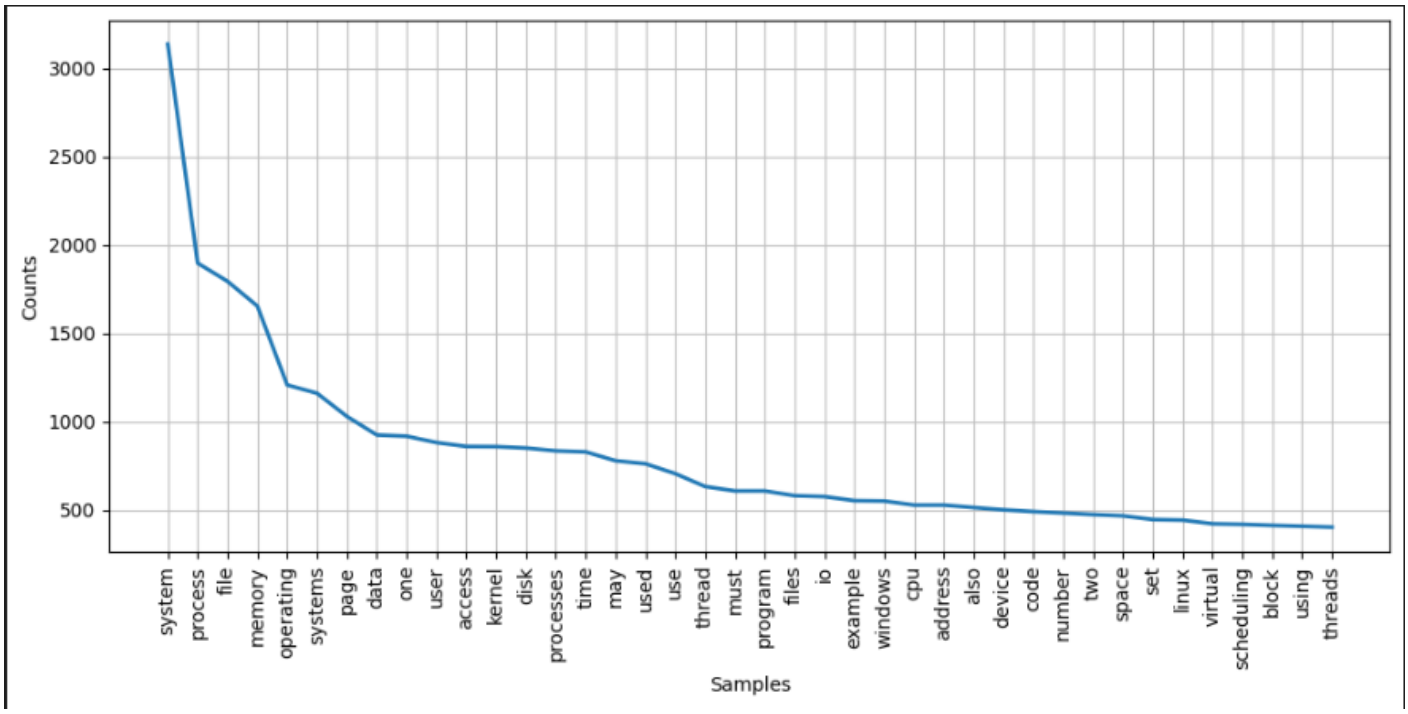
```
# Removing stopwords and storing it into finaltext
stop_words = set(stopwords.words('english'))
tokens = word_tokenize(cleantext)
tokens_final = [i for i in tokens if not i in stop_words] # tokenising with removing stopwords
finaltext = "  "
finaltext = finaltext.join(tokens_final)
```

- Length of finaltext : 13,90,527
- We use **nltk.download("stopwords")** to download the list of stopwords provided by nltk.
- Then, we match each token with the stopword and remove the words which gets matched with the stopwords. This way we create our finaltext.

- **Wordcloud of finaltext (cleantext without stopwords)**



- **Frequency distribution of Finaltext**



➢ **INFERENCE:**

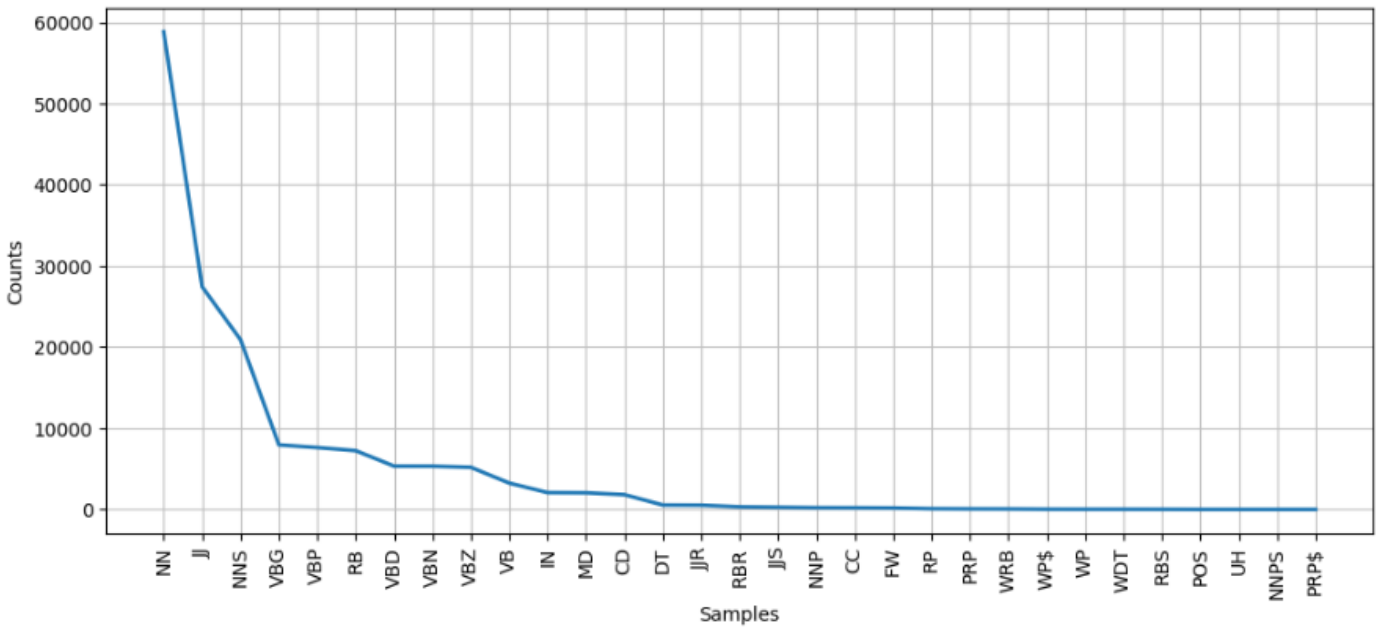
- After the removal of stopwords , we get the finaltext .
- With the help of WordCloud and frequency distribution we can infer what words appears the most in the text.
- As our text is an Operating System Book, we can see that words like **‘System’, ‘process’, ‘file’ and ‘memory’** appears the most.

PoS Tagging

After we thoroughly clean our data, we proceed with the text processing part. Here, we have to assign appropriate Part-of-Speech Tags to the tokens.

For this, we use "**nltk.pos_tag( )**" from the NLTK library of python.

```
from collections import Counter
counts = Counter( tag for word,  tag in tagged)
print(counts)
```

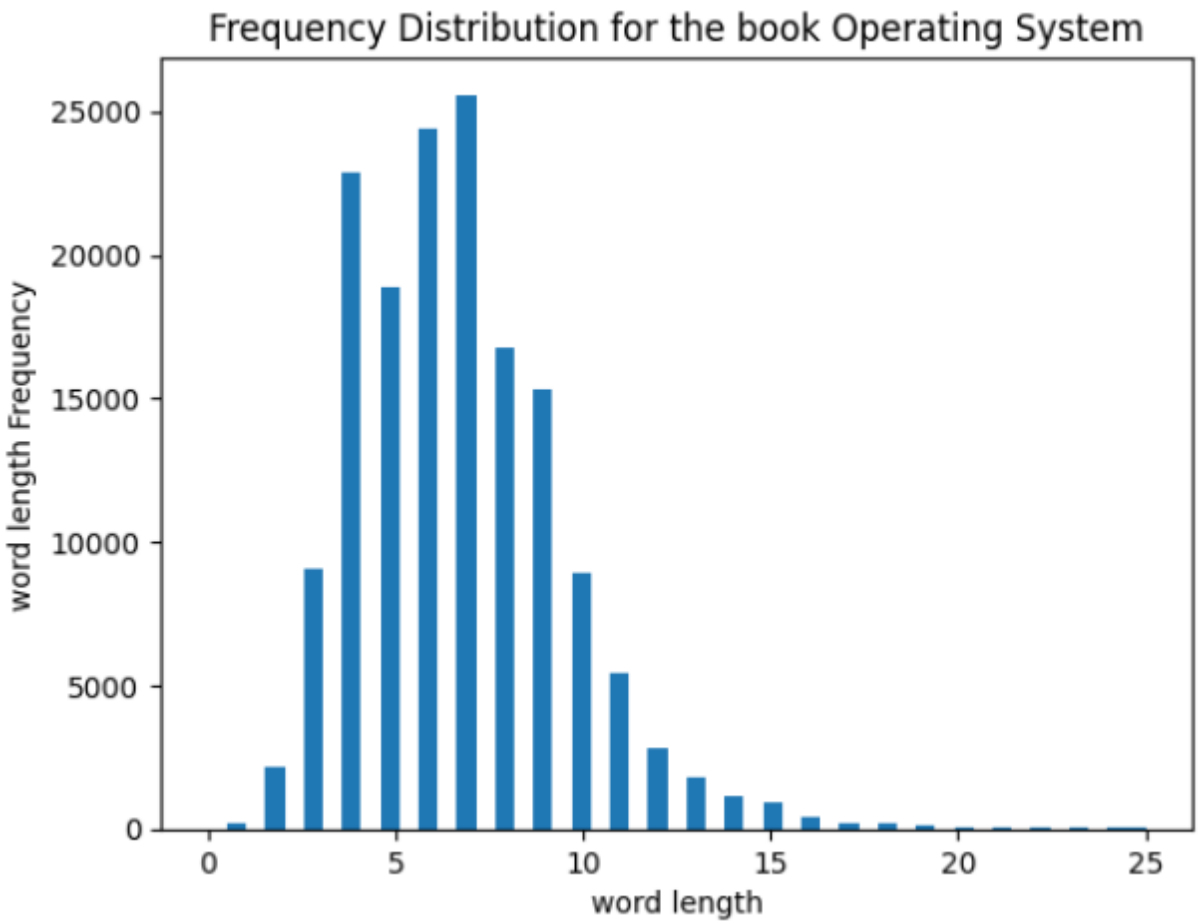


- **Relationship between the word length and frequency:**

Here, we analyze a relationship between the word length and how many words with such word length occurs.

- Using **len( )** function we calculate the length of each token
- Then we plot a graph for frequency of such word lengths using **matplotlib.pyplot.**

➢ **INFERENCE:** We can see that majority of words have their length in Between **4** to **10** letters.



Output for Frequency Distribution for the book Operating Systems

- **The complete code is on our GitHub repository. Link to our repository:**
  Ctrl Alt Elite