Natural Language Processing

CSE-3201

Project Round 1 Report

Course Instructor
Dr. Sakthi Balan Muthiah
Associate Professor, Dept. of CSE

Submitted by:
Naman Jain (20ucs125)
Nipun Garg (20ucs130)
Saksham Jaiswal (20UCS169)
Varun Trivedi (20UCS223)

INDEX

Introduction

Natural language processing (NLP) refers to the branch of computer science and more specifically, the branch of artificial intelligence or AI concerned with giving computers the ability to understand text and spoken words in much the same way human beings can.

NLP combines computational linguistics rule-based modeling of human language with statistical, machine learning, and deep learning models. Together, these technologies enable computers to process human language in the form of text or voice data and to ‘understand’ its full meaning, complete with the speaker or writer’s intent and sentiment.

Objective

In the first round for our project, we will be using the textbook “Software-Engineering-9th-Edition-by-Ian-Sommerville“ and tokenizing its text and then assigning a POS tag to each token.

Firstly we will clean the data and pre-process it. We will then convert the text to tokens and will also create word clouds, we plan to generate two word clouds with and without the stop-words respectively. We will also analyze various plots such as- **token vs frequency distribution** ,**word length vs frequency.**

Finally POS tagging will be done to assign each token its relevant part of speech.

Data Description

The textbook used for this project is a **Computer Science book** (in PDF Format) downloaded in PDF format from the website:
https://engineering.futureuniversity.com
It is then converted into text format using the following online tool
https://www.zamzar.com

Title: **Software Engineering 9th Edition by Ian Sommerville**
No of Words: 289267
No of Characters: 2497996

Importing Libraries

We need to import required libraries in the code to perform relevant operations on our textbook to achieve our objectives.

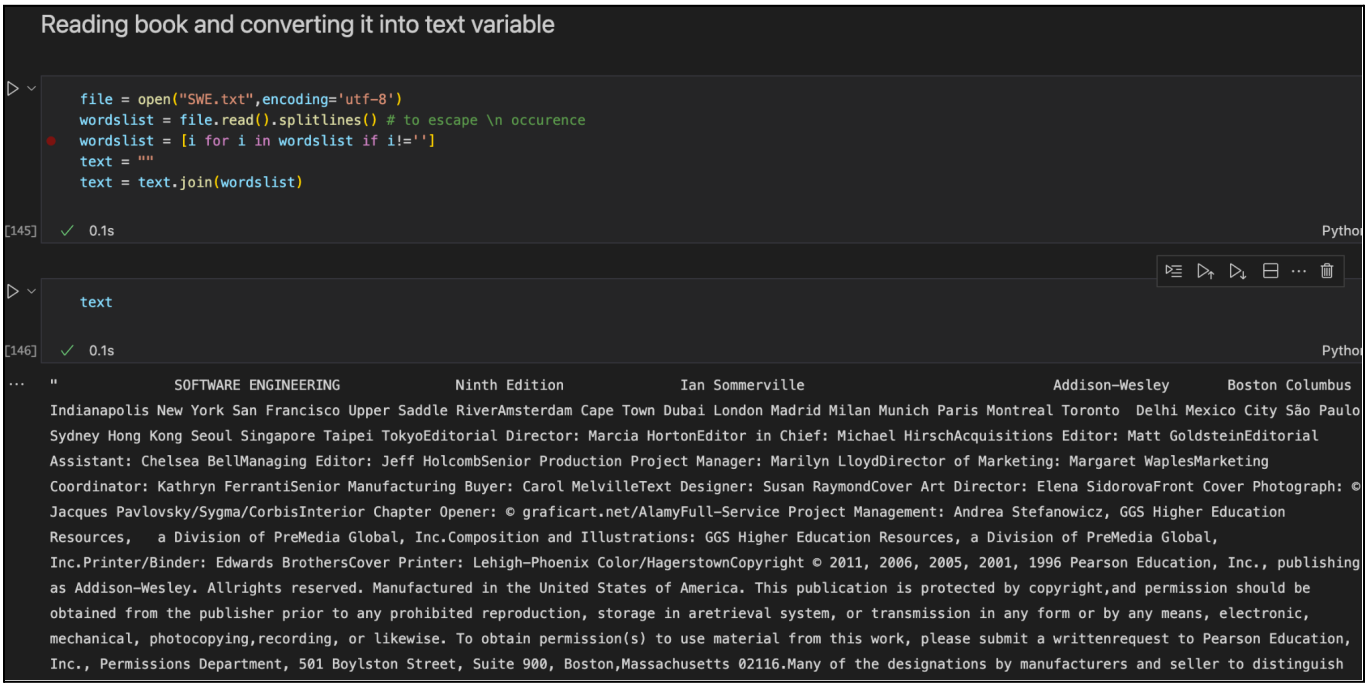
These include **nltk, re, string, wordcloud, matplotlib, pandas, PorterStemmer, word_tokenize, spacy**.

```
import nltk
import re

from nltk.corpus import stopwords
import string
from wordcloud import WordCloud, STOPWORDS
import matplotlib.pyplot as plt
import pandas as pd
from nltk.stem.porter import PorterStemmer
from nltk.tokenize import word_tokenize
import spacy
nlp = spacy.load("en_core_web_sm")
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
```

Importing Datasets

We import our textbook that has already been converted to .txt format.



Text Pre-Processing Steps

Step 1: Convert the downloaded file(.pdf) to editable text file(.txt) using some online tool(www.zamzar.com)

Step 2: Remove Punctuation and special characters from the text such as

' ! ( ) – [ ] { } ■ ; : ' " \ — © • , < > = ` . / ' ' ? ‟ ” @ # $ % ^ & * _ ~ ' ' '

Step 3: Convert all the text to lowercase.

Step 4: Using regular expressions (re library from python) remove and replace URLs, Websites, Line Break characters, Numbers and whitespaces.

```
# remove whitespace from text
from unittest import skip

def remove_whitespace(text):
    return  " ".join(text.split())

#Creating a string which has all the punctuations to be removed
punctuations = '''!()-[]{}■;:'"\–•,<>=./''?“”@#$%^&*_~'''
cleantext = ""
flag=1
for char in text:
    if char not in punctuations:
        cleantext = cleantext + char



#Converting the text into lower case
cleantext = cleantext.lower()
text = cleantext.replace("figure",'')                                                    #replacing figures
cleantext= re.sub(r'(https|http)?:V(\W\.IVI\?]\=1\&|\%) *\b', '', cleantext, flags=re. MULTILINE)   #removing url and links
cleantext= re. sub(r' (www.[a-z]*. [a-z]*)','', cleantext)
cleantext = re.sub(r'[cC]hapter[0-9]+','',cleantext)                                     #removing chapter heading
cleantext =re. sub(r'-(\n)','', cleantext)                                               #adding words segmented by line break
cleantext =re.sub(r' (\n)',' ', cleantext)                                               # changing linebreaks to space
cleantext= re.sub(r'[\+]+','',cleantext)                                                 # removing + & = symbols
cleantext = re.sub(r'[0-9]+','',cleantext)                                               #removing numbers
cleantext= remove_whitespace(cleantext)                                                  #removing whitespace
```

Regular Expressions (a.k.a regex) are a set of pattern matching commands used to detect string sequences in a large text data. These commands are designed to match a family (alphanumeric, digits, words) of text which makes them versatile enough to handle any text / string class.

Output after basic preprocessing of the text:

```
#converting the text into lower case
cleantext = cleantext.lower()
text = cleantext.replace("figure",'')                                    #replacing figures
cleantext= re.sub(r'(https|http)?:V(\W\.IVI\?]\=i\&|\%) *\b', '', cleantext, flags=re. MULTILINE)   #removing url and links
cleantext= re. sub(r' (www.[a-z]*. [a-z]*)','', cleantext)

cleantext = re.sub(r'[cC]hapter[0-9]+','',cleantext)                     #removing chapter heading
cleantext =re. sub(r'-(\n)','', cleantext)                               #adding words segmented by line break
cleantext =re.sub(r' (\n)',' ', cleantext)                               # changing linebreaks to space
cleantext= re.sub(r'[\+]+','',cleantext)                                 # removing + & = symbols
cleantext = re.sub(r'[0-9]+','',cleantext)                               #removing numbers
cleantext= remove_whitespace(cleantext)                                  #removing whitespace
```

236] ✓ 0.8s Python

```
cleantext
```

237] ✓ 0.1s Python

```
'software engineering ninth edition ian sommerville addisonwesley boston columbus indianapolis new york san francisco upper saddle riveramsterdam cape town
dubai london madrid milan munich paris montreal toronto delhi mexico city são paulo sydney hong kong seoul singapore taipei tokyoeditorial director marcia
hortoneditor in chief michael hirschacquisitions editor matt goldsteineditorial assistant chelsea bellmanaging editor jeff holcombsenior production project
manager marilyn lloyddirector of marketing margaret waplesmarketing coordinator kathryn ferrantisenior manufacturing buyer carol melvilletext designer susan
raymondcover art director elena sidorovafront cover photograph jacques pavlovskysygmacorbisinterior chapter opener graficartnetalamyfullservice project
management andrea stefanowicz ggs higher education resources a division of premedia global inccomposition and illustrations ggs higher education resources a
division of premedia global incprinterbinder edwards brotherscover printer lehighphoenix colorhagerstowncopyright pearson education inc publishing as
addisonwesley allrights reserved manufactured in the united states of america this publication is protected by copyrightand permission should be obtained from
the publisher prior to any prohibited reproduction storage in aretrieval system or transmission in any form or by any means electronic mechanical
photocopyingrecording or likewise to obtain permissions to use material from this work please submit a writtenrequest to pearson education inc permissions
department boylston street suite bostonmassachusetts many of the designations by manufacturers and seller to distinguish their products are claimed as
trademarks where those designations appear in this book and the publisher was aware of a trademark claimthe designations have been printed in initial caps or
all capslibrary of congress catalogueinginpublication datasommerville ian software engineering ian sommerville th ed p cm includes index isbn isbn software
engineering i title qas dc —eb— isbn isbn prefaceas i was writing the final chapters in this book in the summer of i realizedthat software engineering was
```

Tokenizing Our Dataset

Next step would be to tokenize our cleaned data. For that we will be using **nltk.word_tokenize library()** which will take our text as input in the form of string and return a list of tokens.

- 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.

```
tokens=word_tokenize(cleantext)
print(len(tokens))
tokens
```

✓ 0.7s

```
291639

Output exceeds the size limit. Open the full output data in a text editor
['software',
 'engineering',
 'ninth',
 'edition',
 'ian',
 'sommerville',
 'addisonwesley',
 'boston',
 'columbus',
 'indianapolis',
 'new',
 'york',
 'san',
 'francisco',
 'upper',
 'saddle',
 'riveramsterdam',
 'cape',
 'town',
 'dubai'
```

Calculating The Frequency Of The Tokens

We use the nltk.FreqDist() function to calculate the individual frequency of the tokens.

```
frequency_distribution=nltk. FreqDist(tokens)
print (frequency_distribution.most_common(15))
freq_dist=list (frequency_distribution)
```

✓ 0.2s Python

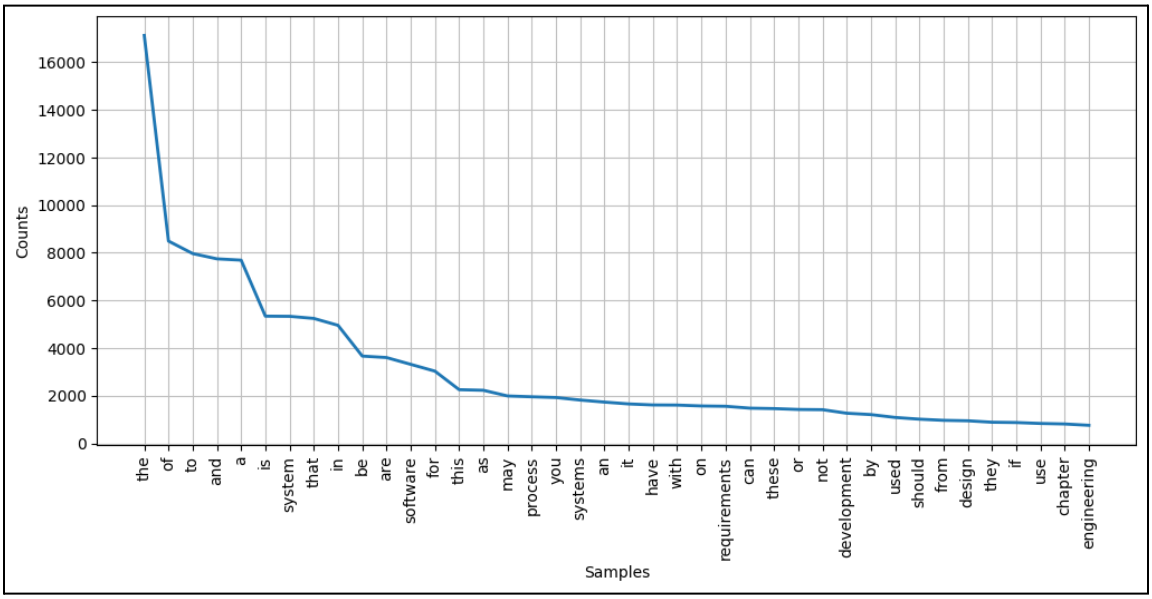
```
[('the', 17121), ('of', 8491), ('to', 7966), ('and', 7745), ('a', 7691), ('system', 5421), ('is', 5339), ('that', 5244), ('in', 4954), ('be', 3664), ('are', 3602), ('software', 3326), ('for', 3031), ('this', 2255), ('as', 2228)]
```

Visualizing Frequency Distribution Of Tokens

We use seaborn plot to analyze the frequency of the tokens on our dataset. We check the first 50 most occurring tokens.

```
tokens = word_tokenize(cleantext)
freq = nltk.FreqDist(tokens)
plt.figure(figsize=(12,5))
freq.plot(50, cumulative=False)
```

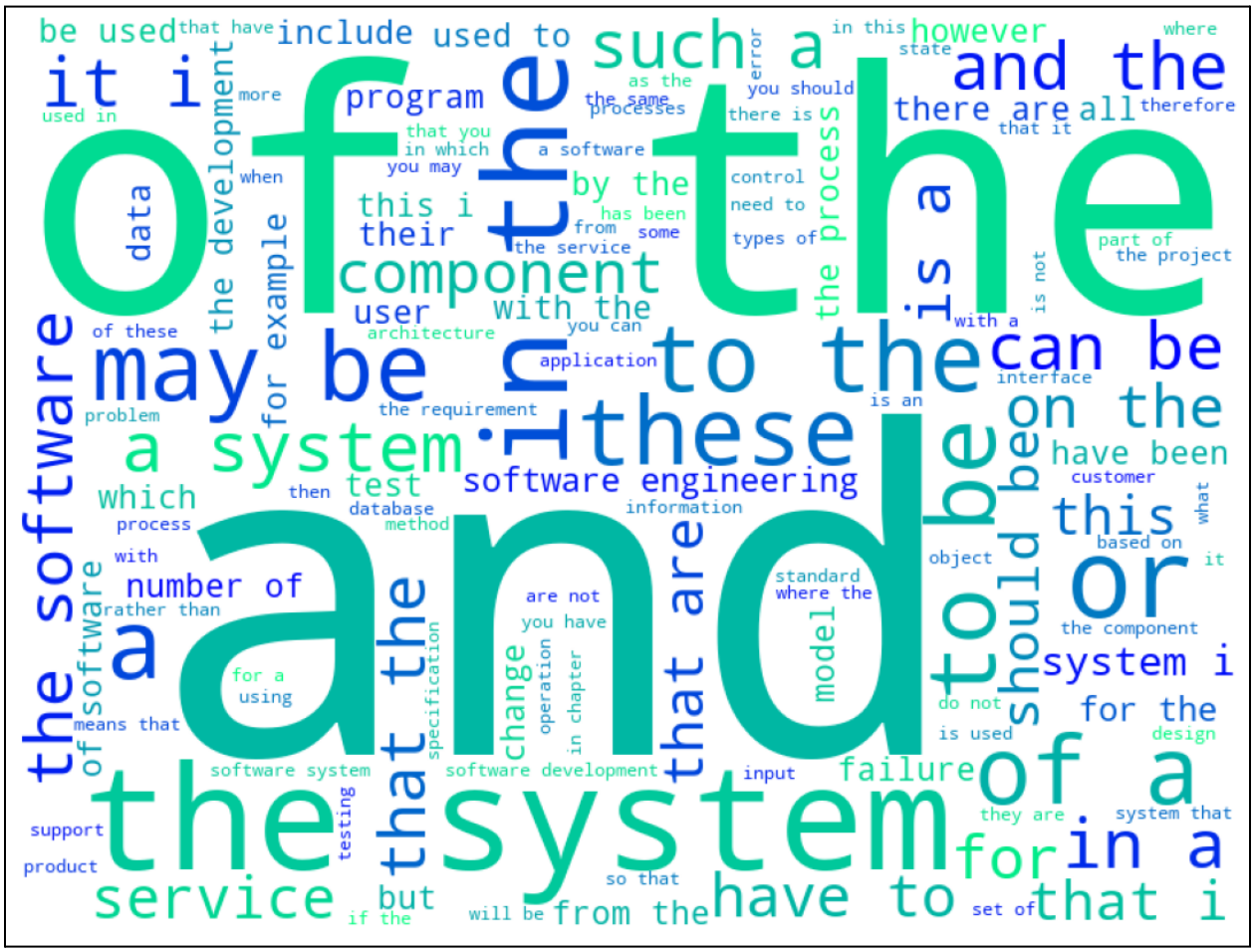
✓ 1.3s



Creating Word Cloud With Stopwords

```
# Word cloud without removing stopwords
wordcloud = WordCloud(width = 800, height = 600,
                      background_color ='white',
                      min_font_size = 10,stopwords = {},colormap='winter').generate(cleantext)

plt.figure(figsize = (12,8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)
plt.show()
```



Word clouds (also known as text clouds or tag clouds) work in a simple way: the more a specific word appears in a source of textual data (such as a speech, blog post, or database), the bigger and bolder it appears in the word cloud.

Uses:

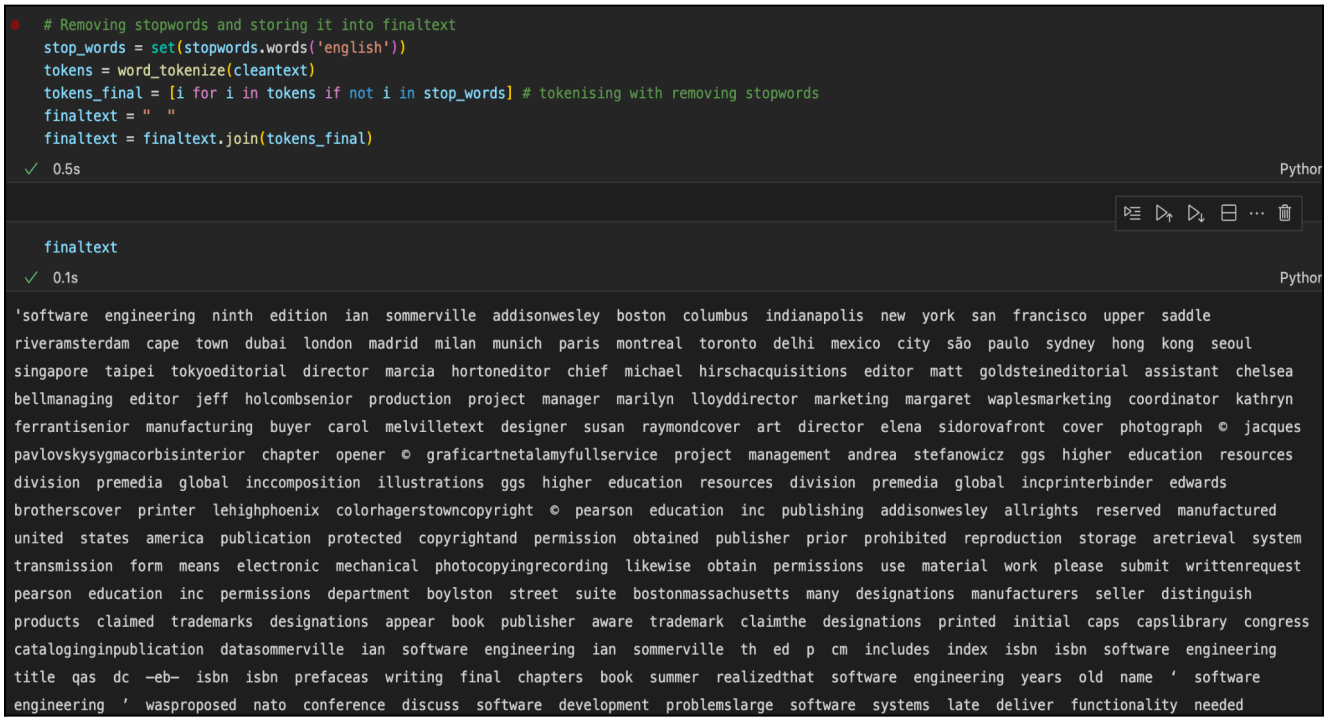
1. Analyzing customer and employee feedback.
2. Identifying new SEO keywords to target.

Removing Stopwords

We will now remove stopwords from our dataset which include but are not limited to **“a”, “the”, “is”, “are”.** Since the English language has a lot of stopwords it would be impossible for us to list them all. So NLTK provides a list of different stopwords used in different languages.

I.) We use nltk.download(‘stopwords’) to download that list.

II.) Then, we match every word/token in the ‘tokens’ variable and if the word is a stopword, we tend to ignore it.



```
# 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)
```

✓ 0.5s Pythor

```
finaltext
```

✓ 0.1s Pythor

```
'software engineering ninth edition ian sommerville addisonwesley boston columbus indianapolis new york san francisco upper saddle
riveramsterdam cape town dubai london madrid milan munich paris montreal toronto delhi mexico city são paulo sydney hong kong seoul
singapore taipei tokyoeditorial director marcia hortoneditor chief michael hirschacquisitions editor matt goldsteineditorial assistant chelsea
bellmanaging editor jeff holcombsenior production project manager marilyn lloyddirector marketing margaret waplesmarketing coordinator kathryn
ferrantisenior manufacturing buyer carol melvilletext designer susan raymondcover art director elena sidorovafront cover photograph © jacques
pavlovskysygmacorbisinterior chapter opener © graficartnetalamyfullservice project management andrea stefanowicz ggs higher education resources
division premedia global inccomposition illustrations ggs higher education resources division premedia global incprinterbinder edwards
brotherscover printer lehighphoenix colorhagerstowncopyright © pearson education inc publishing addisonwesley allrights reserved manufactured
united states america publication protected copyrightand permission obtained publisher prior prohibited reproduction storage aretrieval system
transmission form means electronic mechanical photocopyingrecording likewise obtain permissions use material work please submit writtenrequest
pearson education inc permissions department boylston street suite bostonmassachusetts many designations manufacturers seller distinguish
products claimed trademarks designations appear book publisher aware trademark claimthe designations printed initial caps capslibrary congress
cataloginginpublication datasommerville ian software engineering ian sommerville th ed p cm includes index isbn isbn software engineering
title qas dc –eb– isbn isbn prefaceas writing final chapters book summer realizedthat software engineering years old name ‘ software
engineering ’ waspropposed nato conference discuss software development problemslarge software systems late deliver functionality needed
```

- **Calculating The Frequency Distribution Of New Dataset**

```
frequency_distribution=nltk. FreqDist(tokens)
print (frequency_distribution.most_common(30))
freq_dist=list (frequency_distribution)
```

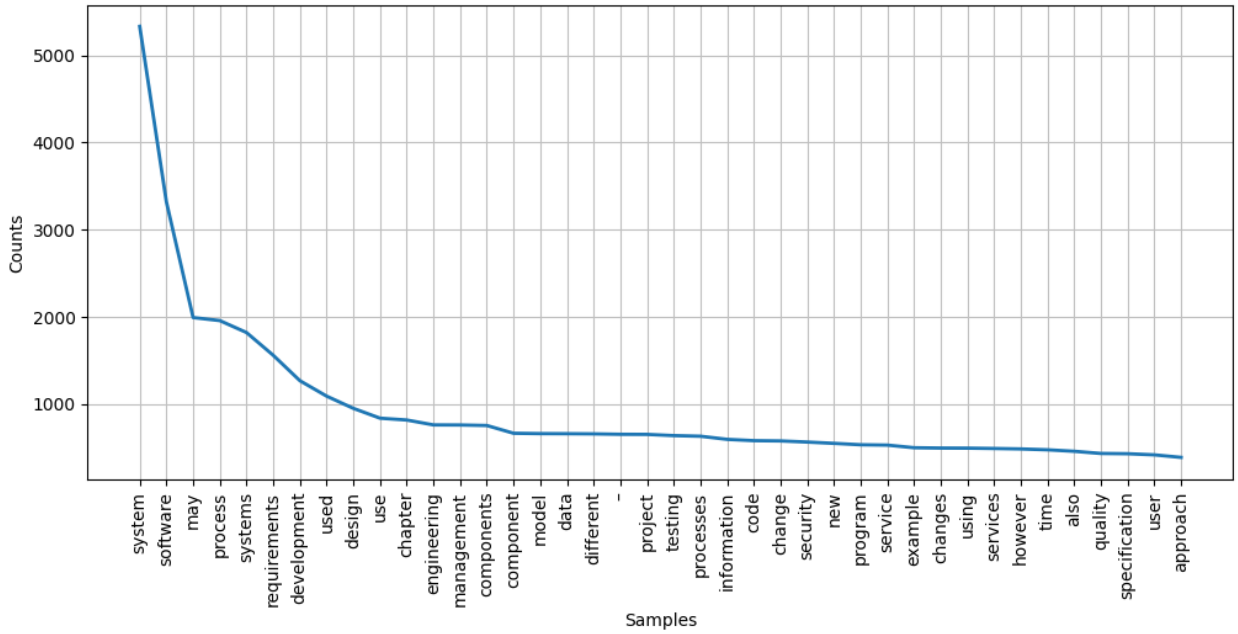
✓ 0.1s Python

```
[('system', 5421), ('software', 3326), ('may', 1990), ('process', 1956), ('systems', 1733), ('requirements', 1557), ('development', 1266), (''', 1255),
('used', 1088), ('design', 948), ('use', 835), ('chapter', 814), ('engineering', 759), ('management', 757), ('components', 748), (''', 687), ('component',
665), ('model', 658), ('data', 657), ('different', 656), ('project', 655), ('—', 650), ('testing', 635), ('processes', 628), ('information', 592), ('code',
577), ('change', 574), ('security', 561), ('new', 546), ('program', 539)]
```

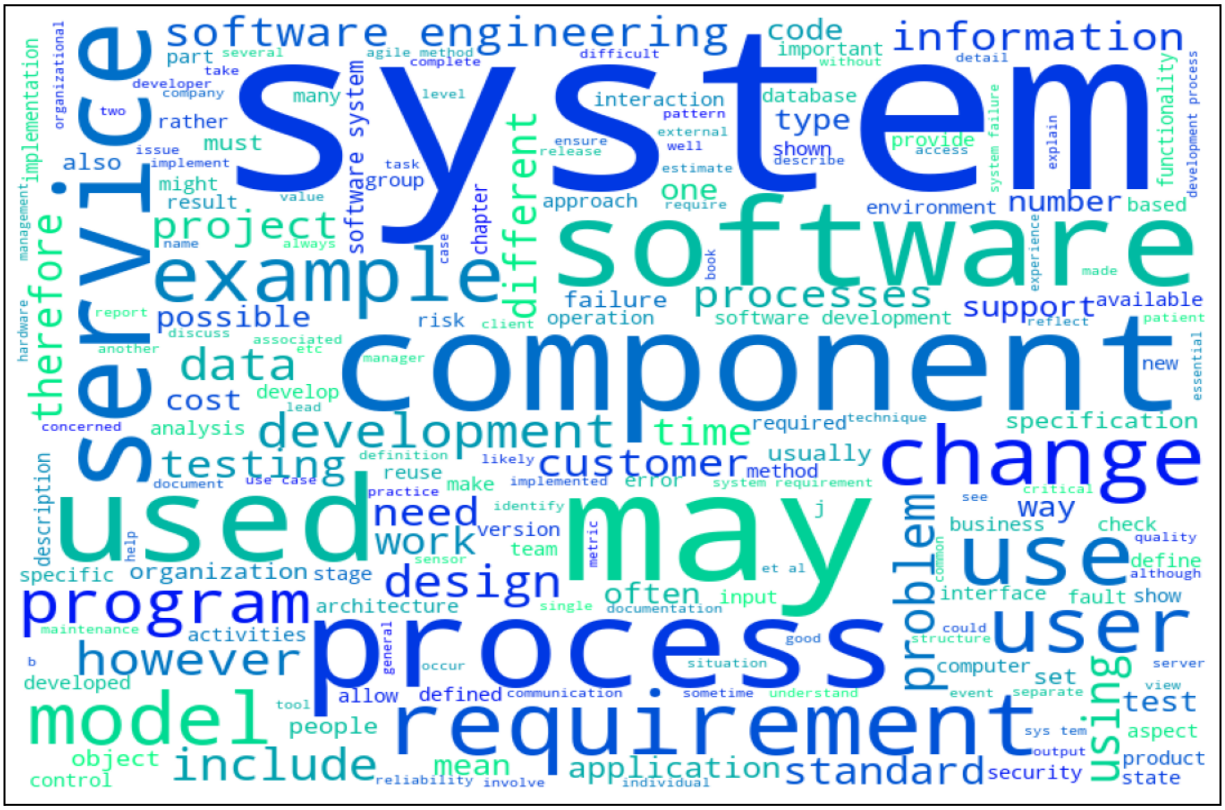
- **Visualizing The Frequency Distribution Of The New Dataset**

```
tokens = word_tokenize(finaltext)
tokens = [i for i in tokens if not i in stop_words]
freq = nltk.FreqDist(tokens)
plt.figure(figsize=(12,5))
freq.plot(40, cumulative=False)
```

✓ 2.3s



- **Creating Word Cloud With Stopwords**



PoS Tagging

Part-of-speech (POS) tagging is a popular Natural Language Processing process which refers to categorizing words in a text (corpus) in correspondence with a particular part of speech, depending on the definition of the word and its context.

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 words.

For this, we use the function “pos_tag( )” from the NLTK library of python.

● The pos_tag() function uses the Penn Treebank Tag Set, which has 36 tags to assign from to the words.
● The pos_tag returns a tuple consisting of the token and the tag.

Here's a snippet of the code and results:

```
tagged = nltk.pos_tag(tokens)
tagged
```

✓ 8.1s

```
('needed', 'VBN'),
('by', 'IN'),
('theirusers', 'NNS'),
('cost', 'RB'),
('more', 'JJR'),
('than', 'IN'),
('expected', 'VBN'),
('and', 'CC'),
('were', 'VBD'),
('unreliable', 'JJ'),
('i', 'NN'),
('did', 'VBD'),
('not', 'RB'),
('attend', 'VB'),
('that', 'IN'),
('conferencebut', 'NN'),
('a', 'DT'),
('year', 'NN'),
('later', 'RB'),
('i', 'JJ'),
('wrote', 'VBD'),
('my', 'PRP$'),
('first', 'JJ'),
('program', 'NN'),
('and', 'CC'),
('started', 'VBD'),
('my', 'PRP$'),
('professional', 'JJ'),
('life', 'NN'),
('in', 'IN'),
('software', 'NN'),
('progress', 'NN'),
('in', 'IN'),
('software', 'NN'),
('engineering', 'NN'),
('has', 'VBZ'),
('been', 'VBN'),
('remarkable', 'JJ'),
('over', 'IN'),
('my', 'PRP$'),
('professional', 'JJ'),
```

```
('topics', 'NNS'),
('during', 'IN'),
('the', 'DT'),
('lifetime', 'NN'),
('of', 'IN'),
('the', 'DT'),
('book', 'NN'),
('material', 'NN'),
('for', 'IN'),
('instructors', 'NNS'),
('the', 'DT'),
('material', 'NN'),
('in', 'IN'),
('this', 'DT'),
('section', 'NN'),
('is', 'VBZ'),
('intended', 'VBN'),
('to', 'TO'),
('support', 'VB'),
('peo', 'JJ'),
('ple', 'NN'),
('who', 'WP'),
('are', 'VBP'),
('teaching', 'VBG'),
('software', 'NN'),
('engineering', 'NN'),
('see', 'VBP'),
('the', 'DT'),
```

```
('abrial', 'JJ'),
('abrial', 'JJ'),
('abrial', 'JJ'),
('absence', 'VBP'),
('absence', 'VBP'),
('absent', 'NN'),
('absolute', 'JJ'),
('absolute', 'JJ'),
('absolute', 'JJ'),
('absolute', 'JJ'),
('absolute', 'JJ'),
('absolute', 'NN'),
('absolute', 'VB'),
('absolutely', 'RB'),
('absolutely', 'RB'),
('absolutes', 'NNS'),
('abstrac', 'JJ'),
('abstrac', 'JJ'),
('abstract', 'JJ'),
('abstract', 'JJ'),
('abstract', 'JJ'),
('abstract', 'JJ'),
```

Distribution of Part-of-Speech Tags in our Data:

Following the application of PoS Tags to relevant words, we attempted to determine the frequency of various tags.

To do this, we first utilized the function of the same name from the library "Counter," which determines the frequency of each tag used in the final text.

We then plotted a graph for the same:

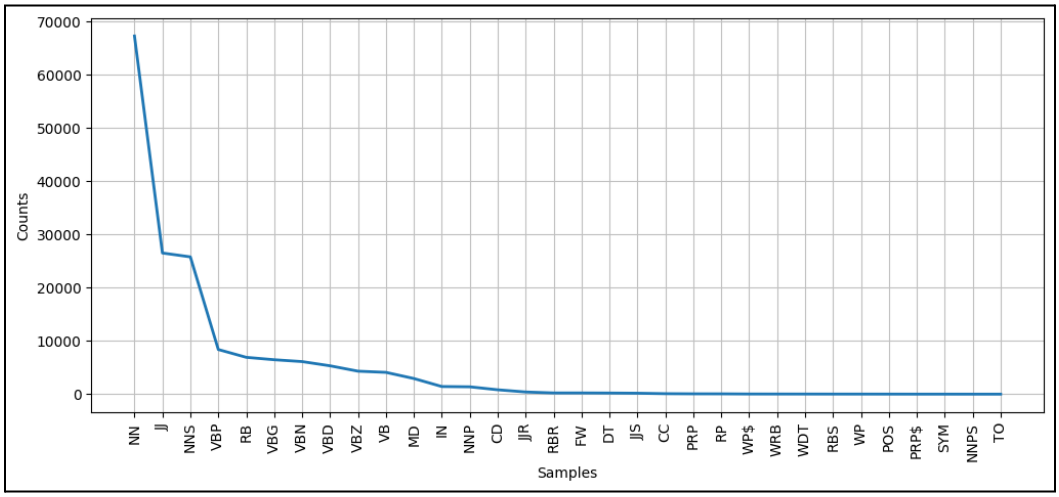
```
from collections import Counter
counts = Counter( tag for word,  tag in tagged)
print(counts)
```

✓ 0.1s Python

```
Counter({'NN': 67265, 'JJ': 26511, 'NNS': 25784, 'VBP': 8383, 'RB': 6918, 'VBG': 6466, 'VBN': 6118, 'VBD': 5325, 'VBZ': 4317, 'VB': 4094, 'MD': 2936, 'IN': 1433, 'NNP': 1380, 'CD': 821, 'JJR': 407, 'RBR': 228, 'FW': 227, 'DT': 209, 'JJS': 176, 'CC': 89, 'PRP': 63, 'RP': 58, 'WP$': 30, 'WRB': 26, 'WDT': 22, 'RBS': 16, 'WP': 15, 'POS': 11, 'PRP$': 5, 'SYM': 4, 'TO': 1, 'NNPS': 1})
```

```
freq_tags = nltk.FreqDist(counts)
plt.figure(figsize=(12,5))
freq_tags.plot(50, cumulative=False)
```

✓ 0.2s



Calculating the lengths of the tokens

```
length=[]
for x in tokens:
    length.append(len(x))
length[:25]
```

✓ 0.6s

```
[8,
 11,
 5,
 7,
 3,
 11,
 13,
 6,
 8,
 12,
 3,
 4,
 3,
 9,
 5,
 6,
 14,
 4,
 4,
 5,
 6,
 6,
 5,
 6,
 5]
```

Calculating the frequency distribution of the length of the words

```
frequency_distribution_length=nltk.FreqDist(length)
frequency_distribution_length
```

✓ 0.1s

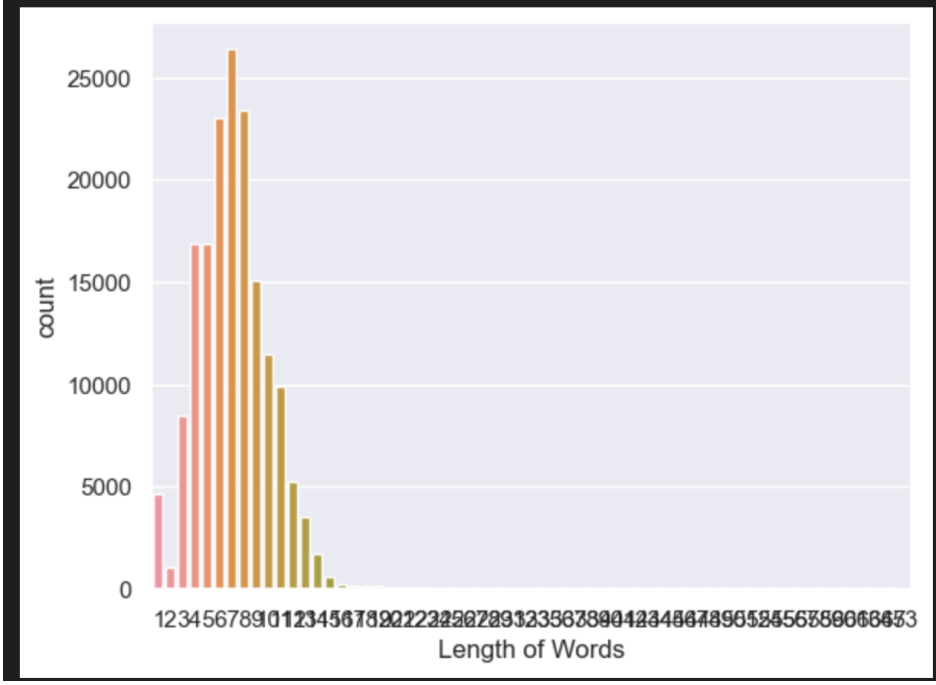
```
FreqDist({7: 26384, 8: 23373, 6: 23001, 4: 16890, 5: 16875, 9: 15049, 10: 11487, 11: 9879, 3: 8450, 12: 5233, ...})
```

Visualizing the relationship of the length of the words and its occurence

```
import seaborn as sns
sns.set(style='darkgrid')

df_len=pd.DataFrame(length)
sns.countplot(x=df_len[0])
plt.xlabel('Length of Words')
plt.show()
```

✓ 0.5s



Relationship between the word length and frequency:

Here, we mainly examine the link between word length and the frequency of words of that length.
Using the "numpy" package, we first associate a bin for the bar graph.
Then using len( ) function we calculate the length of each token
Using matplotlib.pyplot, we then plot a graph to show the frequency of these word lengths.

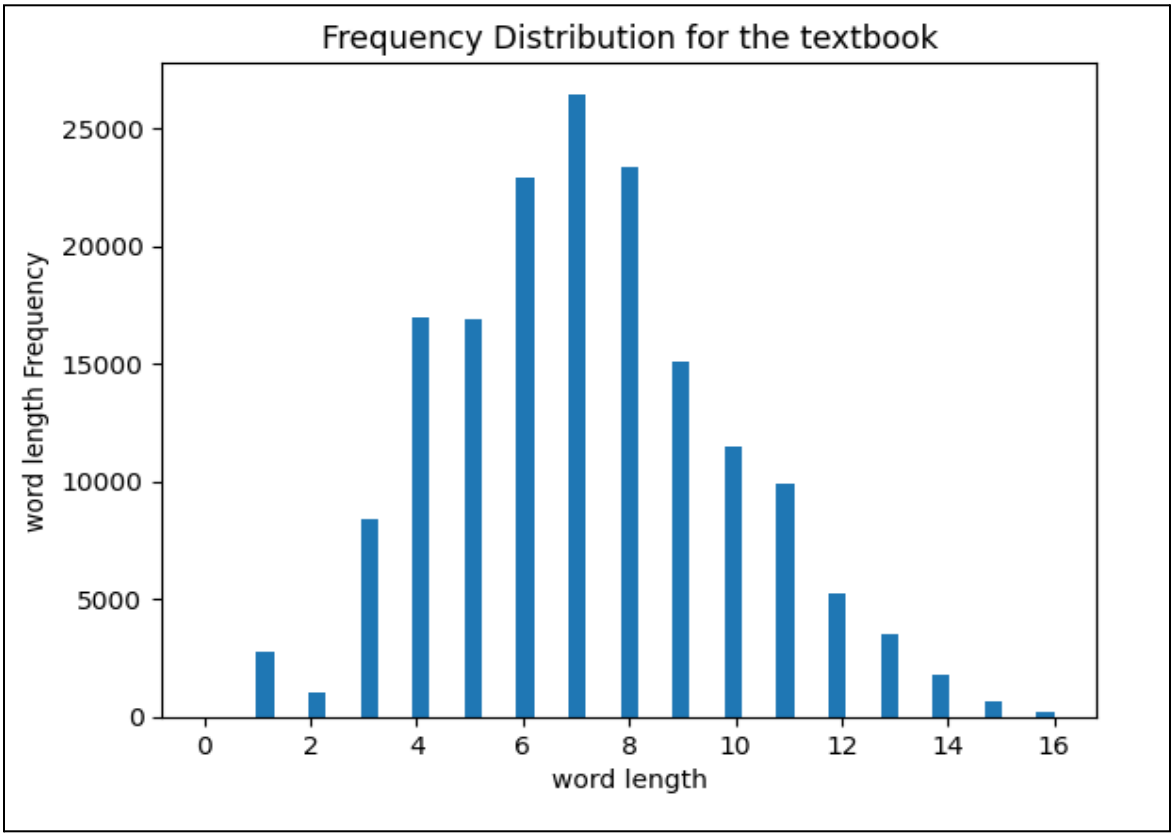
```
For word length vs Frequency distribution
```

```
import numpy as np
bin_size=np.linspace(0,16)
```

✓ 0.2s

```
#Finding Wordlength and storing it as a list
wordLength = [len(r) for r in tokens]

#Plotting histogram of Word length vs Frequency
plt.hist(wordLength, bins=bin_size)
plt.xlabel('word length')
plt.ylabel('word length Frequency')
plt.title('Frequency Distribution for the textbook')
plt.show()
```



GitHub Link:

We've updated the complete code on our GitHub repository . Link for the same is as follows: https://github.com/gnipun05/NLP-Project