NLP PROJECT
TEAM TECH VIBERS

PROJECT TEAM MEMBERS:
1. ARYAN GUPTA(21ucs248)
2. DEEPAK DAYMA(21ucs055)
3. ANSHUL GUPTA(21ucs022)

Table Of Contents:


READING CONTENT OF THE BOOK

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import re
from collections import Counter
from wordcloud import WordCloud
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk import FreqDist
from nltk import pos_tag
from google.colab import files
file_path='/content/Orwell-1949 1984.txt'
with open(file_path, "r") as file:
    content = file.read()
print("TEXT CONTENT OF THE FILE:")
print(content)
```

TEXT CONTENT OF THE FILE:
Eric Arthur Blair (25 June 1903 – 21 January 1950), better known by his pen name George Orwell, was an English novelist and essayist, journalist and critic. His work is characterised by lucid prose, biting social criticism, opposition to totalitarianism, and outspoken support of democratic socialism.
As a writer, Orwell produced literary criticism and poetry, fiction and polemical journalism; and is best known for the allegorical novella Animal Farm (1945) and the dystopian novel Nineteen Eighty-Four (1949). His non-fiction works, including The Road to Wigan Pier (1937),

documenting his experience of working-class life in the north of England, and Homage to Catalonia (1938), an account of his experiences soldiering for the Republican faction of the Spanish Civil War (1936–1939), are as critically respected as his essays on politics and literature, language and culture. In 2008, The Times ranked George Orwell second among "The 50 greatest British writers since 1945".

Orwell's work remains influential in popular culture and in political culture, and the adjective "Orwellian"—describing totalitarian and authoritarian social practices—is part of the English language, like many of his neologisms, such as "Big Brother", "Thought Police", "Two Minutes Hate", "Room 101", "memory hole", "Newspeak", "doublethink", "proles", "unperson", and "thoughtcrime".

Part One

1984

Chapter 1

It was a bright cold day in April, and the clocks were striking thirteen. Winston Smith, his chin nuzzled into his breast in an effort to escape the vile wind, slipped quickly through the glass doors of Victory Mansions, though not quickly enough to prevent a swirl of gritty dust from entering along with him.

The hallway smelt of boiled cabbage and old rag mats. At one end of it a coloured poster, too large for indoor display, had been tacked to the wall. It depicted simply an enormous face, more than a metre wide: the face of a man of about forty-five, with a heavy black moustache and ruggedly handsome features. Winston made for the stairs. It was no use trying the lift. Even at the best of times it was seldom working, and at present the electric current was cut off during daylight hours. It was part of the economy drive in preparation for Hate Week. The flat was seven flights up, and Winston, who was thirty-nine and had a varicose ulcer above his right ankle, went slowly, resting several times on the way. On each landing, opposite the lift-shaft, the poster with the enormous face gazed from the wall. It was one of those pictures which are so contrived that the eyes follow you about when you move. BIG BROTHER IS WATCHING YOU, the caption beneath it ran.

Inside the flat a fruity voice was reading out a list of figFree eBooks at Planet eBook.com

ures which had something to do with the production of
pig-iron. The voice came from an oblong metal plaque like
a dulled mirror which formed part of the surface of the
i h h d
ll Wi
d
i h
d h
i

keyboard_arrow_down New section
LENGTH OF THE BOOK

```
print("Length of the book ",len(content))
print("Data type of the content object is",type(content))
```

Length of the book 596106
Data type of the content object is <class 'str'>

```
pattern1 = re.compile(r'A THOUSAND SPLENDID SUNS', re.IGNORECASE)
pattern2 = r'\*{3}'
pattern3 = re.compile(r'AFTERWORD.*', re.DOTALL)
punctuation_pattern = re.compile(r'[^\w\s]')
page_number_pattern = re.compile(r'\b(?:[1-9]|[1-9]\d|1\d{2}|200|2[0-9][0-9]|300)\b')
chinese_pattern = re.compile(r'[\u4e00-\u9fff]+', re.UNICODE)
txt_without_headers = re.sub(pattern1, '', content)
txt_without_page_numbers = re.sub(page_number_pattern, '', txt_without_headers)
txt_without_chinese_characters = re.sub(chinese_pattern, '', txt_without_page_numbers)
filtered_text = re.sub(pattern2, '', txt_without_chinese_characters)
text_without_punctuation = re.sub(pattern3, '', filtered_text)
final_text = re.sub(punctuation_pattern, '', text_without_punctuation)
print("Modified Text:")
print(final_text)
```

Modified Text:
Eric Arthur Blair June 1903
January 1950 better known by his
pen name George Orwell was an English novelist and essayist journalist
and critic His work is characterised by lucid prose biting social criticism
opposition to totalitarianism and outspoken support of democratic
socialism
As a writer Orwell produced literary criticism and poetry fiction and
polemical journalism and is best known for the allegorical novella
Animal Farm 1945 and the dystopian novel Nineteen EightyFour
1949 His nonfiction works including The Road to Wigan Pier 1937
documenting his experience of workingclass life in the north of England
and Homage to Catalonia 1938 an account of his experiences soldiering
for the Republican faction of the Spanish Civil War 19361939 are as

critically respected as his essays on politics and literature language and culture In 2008 The Times ranked George Orwell second among The greatest British writers since 1945
Orwells work remains influential in popular culture and in political culture and the adjective Orwelliandescribing totalitarian and authoritarian social practicesis part of the English language like many of his neologisms such as Big Brother Thought Police Two Minutes Hate Room memory hole Newspeak doublethink proles unperson and thoughtcrime
Part One

1984
Chapter
I
t was a bright cold day in April and the clocks were striking thirteen Winston Smith his chin nuzzled into his
breast in an effort to escape the vile wind slipped quickly through the glass doors of Victory Mansions though not quickly enough to prevent a swirl of gritty dust from entering along with him
The hallway smelt of boiled cabbage and old rag mats At one end of it a coloured poster too large for indoor display had been tacked to the wall It depicted simply an enormous face more than a metre wide the face of a man of
about fortyfive with a heavy black moustache and ruggedly handsome features Winston made for the stairs It was no use trying the lift Even at the best of times it was seldom working and at present the electric current was cut
off during daylight hours It was part of the economy drive in preparation for Hate Week The flat was seven flights up and Winston who was thirtynine and had a varicose ulcer above his right ankle went slowly resting several times on the way On each landing opposite the liftshaft the poster with the enormous face gazed from the wall It was one of those pictures which are so contrived that the eyes follow you about when you move BIG BROTHER IS WATCHING YOU the caption beneath it ran
Inside the flat a fruity voice was reading out a list of figFree eBooks at Planet eBookcom

ures which had something to do with the production of pigiron The voice came from an oblong metal plaque like a dulled mirror which formed part of the surface of the
i h h d
ll Wi

d
i h
d h
i

```
[ ]  nltk.download('punkt')
     tokens = word_tokenize(final_text)
     T1_frequency_distribution = FreqDist(tokens)
     T1_frequency_distribution_org = T1_frequency_distribution
     T1_frequency_distribution_org
```

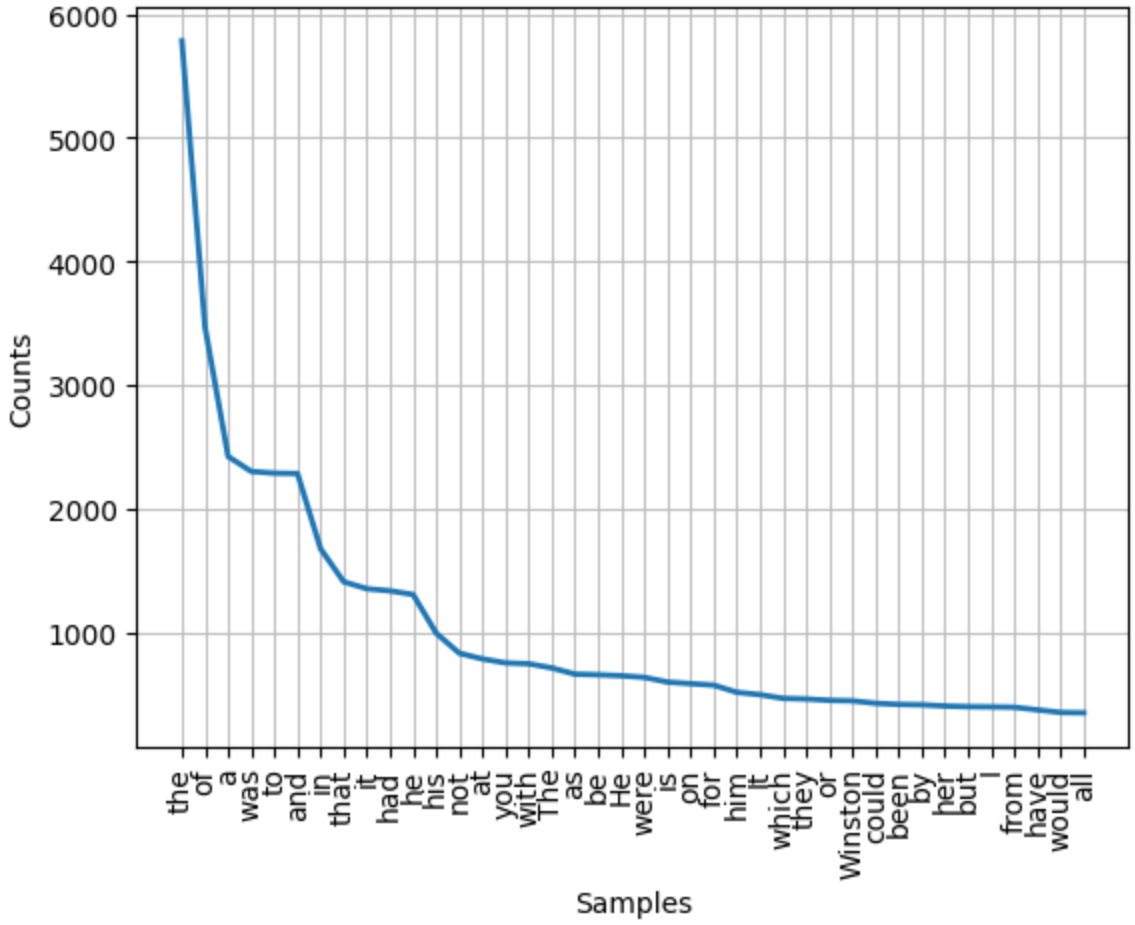
FreqDist({'the': 5784, 'of': 3467, 'a': 2421, 'was': 2300, 'to': 2286, 'and': 2283, 'in': 1674, 'that': 1407, 'it':
1351, 'had': 1334, ...})

```
[ ]  print(tokens)
```

['all', 'four', 'of', 'them', 'simultaneously', 'they', 'were', 'the', 'homes', 'of', 'the', 'four', 'ministries', 'betw

```
[ ]  if "3" in tokens:
       print("yes")
```

PLOT OF THE CHOSEN WORDS BEFORE REMOVING STOPWORDS
T1_frequency_distribution_org.plot(40)




<Axes: xlabel='Samples', ylabel='Counts'>

```
[ ] len(tokens)
```

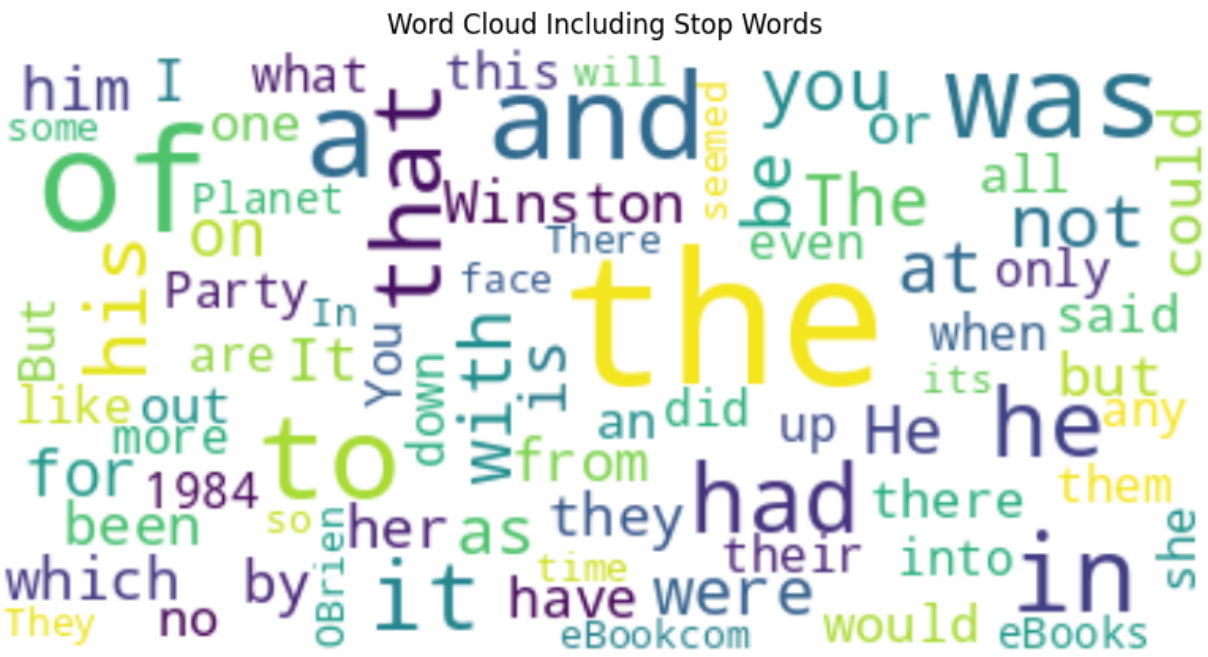
104840

CREATING WORD CLOUD THAT INCLUDES STOPWORDS

```
] import matplotlib.pyplot as plot_
  from matplotlib.pyplot import figure
  from wordcloud import WordCloud

  from collections import Counter
  from collections import OrderedDict

  dictionary = Counter(T1_frequency_distribution)
  cloud = WordCloud(max_font_size = 60, max_words = 80, background_color = "white").generate_from_frequencies(dictionary)
  plot_.figure(figsize = (10, 5))
  plot_.imshow(cloud, interpolation = 'bilinear')
  plot_.axis('off')
  plt.title("Word Cloud Including Stop Words")
  plot_.show()
```




REMOVING STOPWORDS FROM THE BOOK

-> we are basically checking that is the given word present in the stopword collection. If yes then remove it from our data.

```
[ ] nltk.download('stopwords')
    words = word_tokenize(final_text)
    stop_words = set(stopwords.words('english'))
    filtered_words = [word for word in words if word.lower() not in stop_words]
    print("Tokenized Text without Stop Words:")
    print(filtered_words)
```

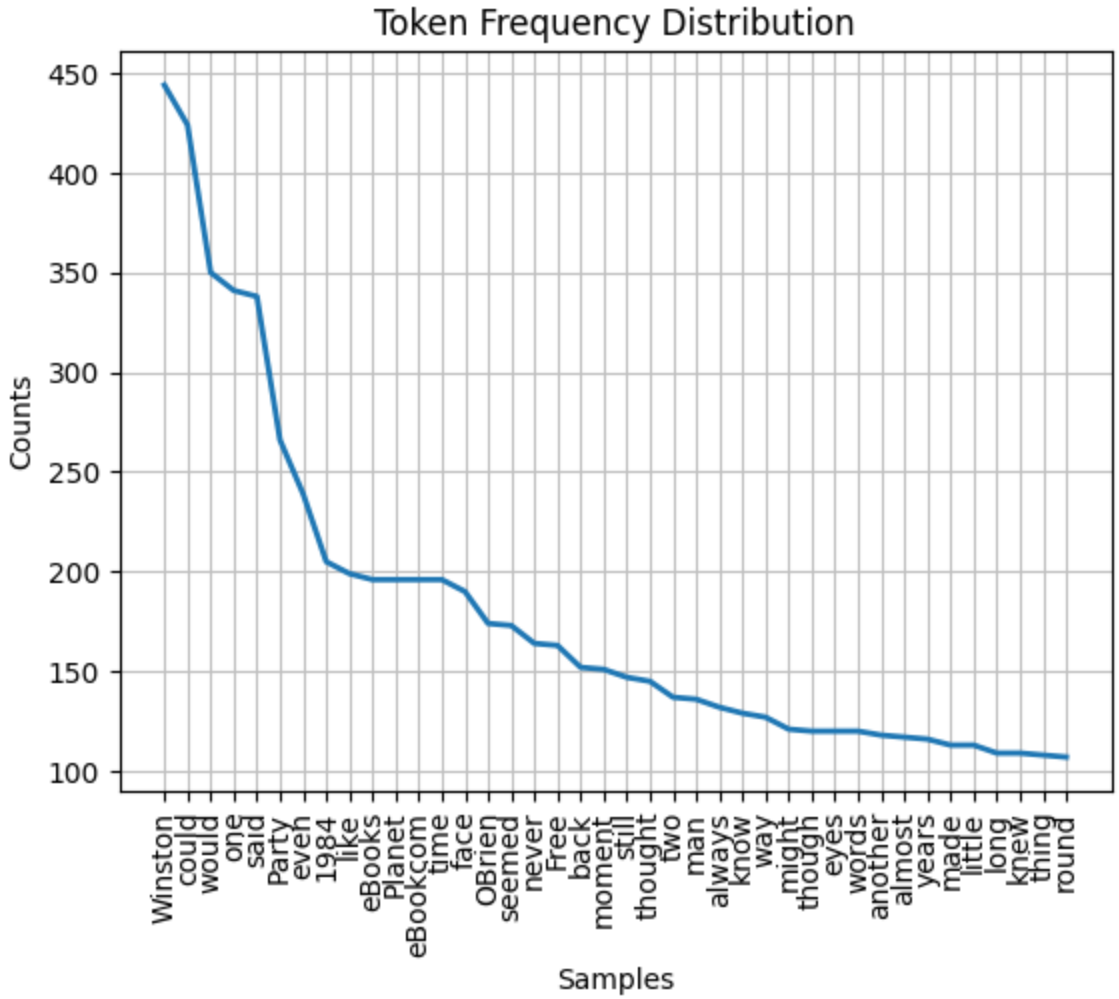
Tokenized Text without Stop Words:

['Eric', 'Arthur', 'Blair', 'June', '1903', 'January', '1950', 'better', 'known', 'pen', 'name', 'George', 'Orwell', 'En

```
[ ] T1_frequency_distribution_withoutstopwords = FreqDist(filtered_words)
    most_common = T1_frequency_distribution_withoutstopwords.most_common(20)
    print("Most common words and their frequencies:")
    for word, freq in most_common:
        print(f"{word}: {freq}")
    plt.title("Token Frequency Distribution")
    T1_frequency_distribution_withoutstopwords.plot(40)
```

```
Most common words and their frequencies:
Winston: 444
could: 424
would: 350
one: 341
said: 338
```

```
Party: 266
even: 239
1984: 205
like: 199
eBooks: 196
Planet: 196
eBookcom: 196
time: 196
face: 190
OBrien: 174
seemed: 173
never: 164
Free: 163
back: 152
moment: 151
```



After removing the stop words in the previous sections, we are using matplotlib library to map a frequency plot of the words that we have. It can be observed that the word "WINSTON" is the two most frequent words.

```
dictionary = Counter(filtered_words)
cloud = WordCloud(max_font_size = 60, max_words = 80, background_color = "white").generate_from_frequencies(dictionary)
plot_.figure(figsize = (10,5))
plot_.imshow(cloud, interpolation = 'bilinear')
plot_.axis('off')
plt.title("Word Cloud after removing stopwords")
plot_.show()
```

```
[ ] nltk.download('averaged_perceptron_tagger')
```

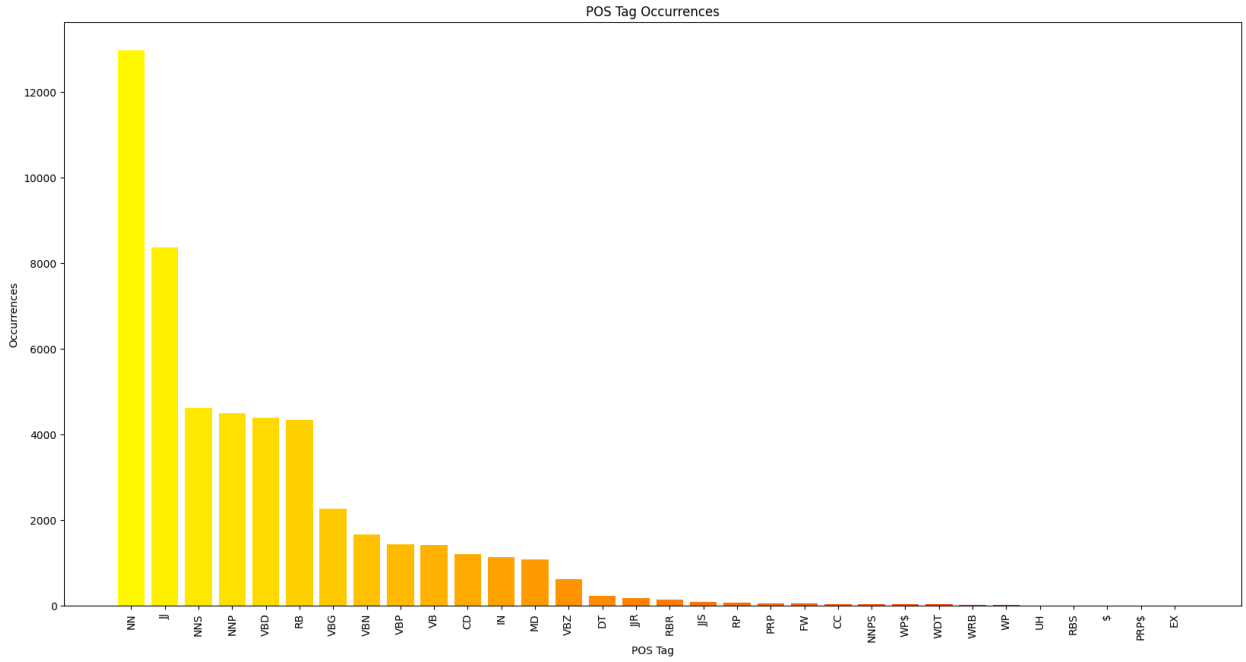
```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
True
```

```
pos_tagged_words = pos_tag(filtered_words)
pos_tagged_words[:15]
```

[('Eric', 'NNP'),
 ('Arthur', 'NNP'),
 ('Blair', 'NNP'),
 ('June', 'NNP'),
 ('1903', 'CD'),
 ('January', 'NNP'),
 ('1950', 'CD'),
 ('better', 'JJR'),
 ('known', 'VBN'),
 ('pen', 'JJ'),
 ('name', 'NN'),
 ('George', 'NNP'),
 ('Orwell', 'NNP'),
 ('English', 'NNP'),
 ('novelist', 'NN')]

-> AS WINSTON IS THE MOST OCCURED WORD IN OUR NOVEL, THE OCCURENCE OF "NN" postag should be highest. Let's observe that by
plotting the POS-tag plot.

```
tag_counts = Counter(tag for word, tag in pos_tagged_words)
tags, counts = zip(*sorted(tag_counts.items(), key=lambda x: x[1], reverse=True))
plt.figure(figsize=(20, 10))
plt.xlabel('POS Tag')
plt.ylabel('Occurrences')
plt.title('POS Tag Occurrences')
plt.xticks(rotation=90)
plt.bar(tags, counts, color=sns.color_palette("autumn_r",len(tags)))
```




<BarContainer object of 32 artists>

```
start_index = tokens.index('one') + 1
end_index = tokens.index('two', start_index)
selected_words = tokens[start_index:end_index]
selected_words[:35]
```

['end',
'of',
'it',
'a',
'coloured',
'poster',
'too',
'large',
'for',
'indoor',
'display',
'had',
'been',
'tacked',
'to',

'the',
'wall',
'It',
'depicted',
'simply',
'an',
'enormous',
'face',
'more',
'than',
'a',
'metre',
'wide',
'the',
'face',
'of',
'a',
'man',
'of',
'about']

```
[ ]  len(selected_words)
```

## BIGRAM MODELLING ON CHAPTER 1

```
[ ]  Chapter1to2 = selected_words
     from nltk.util import bigrams
     bi_grams = list(bigrams(Chapter1to2))
     bigrams_frequency = nltk.FreqDist(bi_grams)
     cfd = nltk.ConditionalFreqDist(bi_grams)
```

```
     unique_words = list(set(word for bigram in bigrams_frequency for word in bigram))
     bigram_matrix = pd.DataFrame(0, columns=unique_words, index=unique_words, dtype=float)
     word_freq = nltk.FreqDist(Chapter1to2)
     bigram_probabilities = {}
     for word1 in unique_words:
       for word2 in unique_words:
         conditional_freq = cfd[word1][word2]
         first_word_count = word_freq[word1]
         probability = float(conditional_freq)/(first_word_count)
         bigram_probabilities[word1, word2] = probability
         bigram_matrix.at[word1, word2] = probability

     bigram_matrix = bigram_matrix.fillna(0)
     print(bigram_matrix)
```

```
           leaving   So   of  slogans  still  moment  itself  packet
darted  \
leaving        0.0  0.0  0.0      0.0    0.0     0.0     0.0     0.0
0.0
So             0.0  0.0  0.0      0.0    0.0     0.0     0.0     0.0
0.0
of             0.0  0.0  0.0      0.0    0.0     0.0     0.0     0.0
0.0
slogans        0.0  0.0  1.0      0.0    0.0     0.0     0.0     0.0
```

```
0.0
still           0.0  0.0  0.0      0.0    0.0     0.0     0.0     0.0
0.0
...             ...  ...  ...      ...    ...     ...     ...     ...
...
livingroom      0.0  0.0  0.0      0.0    0.0     0.0     0.0     0.0
0.0
slummy          0.0  0.0  0.0      0.0    0.0     0.0     0.0     0.0
0.0
between         0.0  0.0  0.0      0.0    0.0     0.0     0.0     0.0
0.0
colourless      0.0  0.0  0.0      0.0    0.0     0.0     0.0     0.0
0.0
Scattered       0.0  0.0  0.0      0.0    0.0     0.0     0.0     0.0
0.0

            peculiarly  ...  again  see  blank  live   pigiron  livingroom
\
leaving            0.0  ...    0.0  0.0    0.0   0.0  0.000000         0.0
So                 0.0  ...    0.0  0.0    0.0   0.0  0.000000         0.0
of                 0.0  ...    0.0  0.0    0.0   0.0  0.013889         0.0
slogans            0.0  ...    0.0  0.0    0.0   0.0  0.000000         0.0
still              0.0  ...    0.0  0.0    0.0   0.0  0.000000         0.0
...                ...  ...    ...  ...    ...   ...       ...         ...
livingroom         0.0  ...    0.0  0.0    0.0   0.0  0.000000         0.0
slummy             0.0  ...    0.0  0.0    0.0   0.0  0.000000         0.0
between            0.0  ...    0.0  0.0    0.0   0.0  0.000000         0.0
colourless         0.0  ...    0.0  0.0    0.0   0.0  0.000000         0.0
Scattered          0.0  ...    0.0  0.0    0.0   0.0  0.000000         0.0

            slummy  between  colourless  Scattered
leaving        0.0      0.0    0.000000        0.0
So             0.0      0.0    0.000000        0.0
of             0.0      0.0    0.013889        0.0
slogans        0.0      0.0    0.000000        0.0
still          0.0      0.0    0.000000        0.0
...            ...      ...         ...        ...
livingroom     0.0      0.0    0.000000        0.0
slummy         0.0      0.0    0.000000        0.0
between        0.0      0.0    0.000000        0.0
colourless     0.0      0.0    0.000000        0.0
Scattered      0.0      0.0    0.000000        0.0
```

-> ABOVE ARE THE BIGRAM PROBABILITIES

```
top_ten_bigrams = sorted(bigram_probabilities.items(), key=lambda x: x[1], reverse=True)[:15]
print("Top Ten Bigrams with Maximum Probability:")
```

```
for bigram, probability in top_ten_bigrams:
word1, word2 = bigram
print(f"{word1} -> {word2}: Probability = {probability:.4f}")
```
Top Ten Bigrams with Maximum Probability:
leaving -> the: Probability = 1.0000
So -> completely: Probability = 1.0000
slogans -> of: Probability = 1.0000
packet -> marked: Probability = 1.0000



darted -> away: Probability = 1.0000
peculiarly -> beautiful: Probability = 1.0000
flight -> It: Probability = 1.0000
A -> kilometre: Probability = 1.0000
depicted -> simply: Probability = 1.0000
distinguishable -> The: Probability = 1.0000
fortyfive -> with: Probability = 1.0000
after -> terrace: Probability = 1.0000
died -> down: Probability = 1.0000
town -> just: Probability = 1.0000
only -> by: Probability = 1.0000

-> ABOVE ARE THE PROBABILITIES OF TOP TEN BIGRAMS

keyboard_arrow_down SHANNON GAME ON CHAPTER 2

```
def make_guess(previous_word, bigram_probabilities):
later_probabilities = {word: prob for (prev, word), prob in bigram_probabilities.items() if prev == previous_word}
optimal_guesses = sorted(later_probabilities, key=later_probabilities.get, reverse=True)
return optimal_guesses

def play_shannons_game(existing_string, bigram_probabilities):
s=" "
print("Welcome to Shannon's Game!")
print("Think of a word, and I will try to guess it based on the provided string.")
```

```
print("Please respond with 'yes' or 'no' to my guesses.")
print("You can end the game by typing 'exit'.")
previous_word = input("Think of a starting word: ")
print(f"Starting word: {previous_word}")
s=s+" "+previous_word
while True:
guesses = make_guess(previous_word, bigram_probabilities)
for guess in guesses:
response = input(f"Is it '{guess}'? (yes/no): ")
if response == 'yes':
previous_word = guess
s=s+" "+guess
print(f"Sentence: {s}")
break
elif response=='no':
continue
elif response == 'exit':
print("Thanks for playing!")
return
else:
print("No more guesses. Thanks for playing!")
return
start_index = tokens.index('two') + 1
end_index = tokens.index('three', start_index)
chapter_2 = tokens[start_index:end_index]
chapter_2[:15]
['dollars',
'fifty',
'At',
'the',
'time',
'he',
'was',
'not',
'conscious',
'of',
'wanting',
'it',
'for',
'any',
'particular']
len(chapter_2)
755
```

play_shannons_game(chapter_2, bigram_probabilities)

https://colab.research.google.com/drive/1C_Jn6dkrVbEbj5X9cXaRkRabU6OFm_6P?usp=sharing#scrollTo=RYNUB9dxUFeq&printMode=true

9/10

18/12/2023, 23:04

TECH_VIBERS.ipynb - Colaboratory

Welcome to Shannon's Game!
Think of a word, and I will try to guess it based on the provided string.
Please respond with 'yes' or 'no' to my guesses.
You can end the game by typing 'exit'.
Think of a starting word: flight
Starting word: flight
Is it 'It'? (yes/no): no
Is it 'leaving'? (yes/no): yes
Sentence:
flight leaving
Is it 'the'? (yes/no): no
Is it 'leaving'? (yes/no): no
Is it 'So'? (yes/no): no
Is it 'of'? (yes/no): no
Is it 'slogans'? (yes/no): no
Is it 'still'? (yes/no): no
Is it 'moment'? (yes/no): no
Is it 'itself'? (yes/no): yes
Sentence:
flight leaving itself
Is it 'with'? (yes/no): exit
Thanks for playing!

GITHUB LINK:
https://github.com/algoviber/TECH_VIBERS_NLP_PROJECT_1/tree/main

https://colab.research.google.com/drive/1C_Jn6dkrVbEbj5X9cXaRkRabU6OFm_6P?usp=sharing#scrollTo=RYNUB9dxUFeq&printMode=true

10/10

```
import nltk
import pandas as pd
import numpy as np
import re
import string
import matplotlib.pyplot as plt
import numpy as np
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.corpus import stopwords
from nltk.probability import FreqDist
from nltk.tag import pos_tag
from nltk import FreqDist
from nltk.util import bigrams
from wordcloud import WordCloud
```

```
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
```

```
file = open(r"/content/Orwell-1949 1984.txt",encoding='utf-8')
wordslist = file.read().splitlines() # to escape \n occurence
wordslist = [i for i in wordslist if i!='']
text = ""
text = text.join(wordslist)
```

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

#Converting the text into lower case
plaintext = plaintext.lower()
 #Tokenize
tokens = word_tokenize(plaintext)
```

```
# Here we use spacy model
import spacy
from sklearn.metrics import precision_score, recall_score, f1_score
# Load spaCy English language model
nlp = spacy.load("en_core_web_sm")
```

```
text1 = """to pieces. The plaster flaked constantly from ceilings and
walls, the pipes burst in every hard frost, the roof leaked
whenever there was snow, the heating system was usually
running at half steam when it was not closed down altogether from motives of economy. Repairs, except what you
could do for yourself, had to be sanctioned by remote committees which were liable to hold up even the mending of a
window-pane for two years.
'Of course it's only because Tom isn't home,' said Mrs
Parsons vaguely.
The Parsons' flat was bigger than Winston's, and dingy
in a different way. Everything had a battered, trampled-on
look, as though the place had just been visited by some large
violent animal. Games impedimenta—hockey-sticks, boxing-gloves, a burst football, a pair of sweaty shorts turned
inside out—lay all over the floor, and on the table there was
a litter of dirty dishes and dog-eared exercise-books. On
the walls were scarlet banners of the Youth League and the
Spies, and a full-sized poster of Big Brother. There was the
usual boiled-cabbage smell, common to the whole building,
but it was shot through by a sharper reek of sweat, which—
one knew this at the first sniff, though it was hard to say
how—was the sweat of some person not present at the moment. In another room someone with a comb and a piece of
toilet paper was trying to keep tune with the military music
which was still issuing from the telescreen.
'It's the children,' said Mrs Parsons, casting a half-apprehensive glance at the door. 'They haven't been out today.
And of course——'"""

text2 = """garden walls sagging in all directions? And the bombed
sites where the plaster dust swirled in the air and the willow-herb straggled over the heaps of rubble; and the places
where the bombs had cleared a larger patch and there had
sprung up sordid colonies of wooden dwellings like chicken-houses? But it was no use, he could not remember:
nothing remained of his childhood except a series of brightlit tableaux occurring against no background and mostly
unintelligible.
The Ministry of Truth—Minitrue, in Newspeak [Newspeak was the official language of Oceania. For an account
of its structure and etymology see Appendix.]—was startlingly different from any other object in sight. It was an
enormous pyramidal structure of glittering white concrete, soaring up, terrace after terrace, 300 metres into the
air. From where Winston stood it was just possible to read,
picked out on its white face in elegant lettering, the three
slogans of the Party:
WAR IS PEACE
FREEDOM IS SLAVERY
IGNORANCE IS STRENGTH
The Ministry of Truth contained, it was said, three
thousand rooms above ground level, and corresponding
ramifications below. Scattered about London there were
just three other buildings of similar appearance and size.
So completely did they dwarf the surrounding architecture that from the roof of Victory Mansions you could see"""

text3 = """all four of them simultaneously. They were the homes of
the four Ministries between which the entire apparatus
of government was divided. The Ministry of Truth, which
concerned itself with news, entertainment, education, and
the fine arts. The Ministry of Peace, which concerned itself
with war. The Ministry of Love, which maintained law and
order. And the Ministry of Plenty, which was responsible
for economic affairs. Their names, in Newspeak: Minitrue,
Minipax, Miniluv, and Miniplenty.
The Ministry of Love was the really frightening one.
There were no windows in it at all. Winston had never been
inside the Ministry of Love, nor within half a kilometre of it.
It was a place impossible to enter except on official business,
and then only by penetrating through a maze of barbedwire entanglements, steel doors, and hidden machine-gun
nests. Even the streets leading up to its outer barriers were
roamed by gorilla-faced guards in black uniforms, armed
with jointed truncheons.
Winston turned round abruptly. He had set his features
into the expression of quiet optimism which it was advisable to wear when facing the telescreen. He crossed the
room into the tiny kitchen. By leaving the Ministry at this
time of day he had sacrificed his lunch in the canteen, and
he was aware that there was no food in the kitchen except
a hunk of dark-coloured bread which had got to be saved
for tomorrow's breakfast. He took down from the shelf a
bottle of colourless liquid with a plain white label marked
VICTORY GIN. It gave off a sickly, oily smell, as of Chinese
rice-spirit. Winston poured out nearly a teacupful, nerved"""
```

```python
# Assuming you have the three texts stored in text1, text2, and text3

texts = [text1, text2, text3]

recognized_entities_all = []
recognized_entity_types_all = []

for text in texts:
    doc = nlp(text)

    # Extract recognized entities
    recognized_entities = {ent.text: (ent.start_char, ent.end_char) for ent in doc.ents}
    recognized_entities_all.append(recognized_entities)

    # Recognize entity types
    recognized_entity_types = {ent.text: ent.label_ for ent in doc.ents}
    recognized_entity_types_all.append(recognized_entity_types)

# Now, recognized_entities_all and recognized_entity_types_all contain the results for each document
# Each element in these lists corresponds to the entities and entity types for the respective document.
```

```python
for i, (entities, entity_types) in enumerate(zip(recognized_entities_all, recognized_entity_types_all), start=1):
    print(f"Document {i}:")

    print("Recognized Entities:")
    for entity, (start, end) in entities.items():
        print(f"  - {entity}: Start: {start}, End: {end}")

    print("\n")
```

```
Document 1:
Recognized Entities:
  - half: Start: 1473, End: 1477
  - two years: Start: 418, End: 427
  - Tom: Start: 458, End: 461
  - Mrs
Parsons: Start: 480, End: 491
  - Winston: Start: 535, End: 542
  - the Youth League: Start: 944, End: 960
  - Spies: Start: 969, End: 974
  - one: Start: 1147, End: 1150
  - first: Start: 1168, End: 1173
  - telescreen: Start: 1412, End: 1422
  - Mrs Parsons: Start: 1450, End: 1461
  - today: Start: 1534, End: 1539


Document 2:
Recognized Entities:
  - colonies: Start: 248, End: 256
  - The Ministry of Truth: Start: 1003, End: 1024
  - Newspeak: Start: 506, End: 514
  - Oceania: Start: 554, End: 561
  - 300 metres: Start: 786, End: 796
  - Winston: Start: 822, End: 829
  - three: Start: 1168, End: 1173
  - Party: Start: 942, End: 947
  - IGNORANCE: Start: 981, End: 990
```

```
  - thousand: Start: 1055, End: 1063
  - London: Start: 1145, End: 1151


Document 3:
Recognized Entities:
  - four: Start: 60, End: 64
  - The Ministry of Truth: Start: 138, End: 159
  - The Ministry of Peace: Start: 240, End: 261
  - The Ministry of Love: Start: 492, End: 512
  - the Ministry of Plenty: Start: 354, End: 376
  - Newspeak: Start: 438, End: 446
  - Minipax: Start: 458, End: 465
  - Miniluv: Start: 467, End: 474
  - Miniplenty: Start: 480, End: 490
  - Winston: Start: 1575, End: 1582
  - the Ministry of Love: Start: 611, End: 631
  - half a kilometre: Start: 644, End: 660
  - tomorrow: Start: 1389, End: 1397
  - Chinese: Start: 1554, End: 1561
```

```python
all_dataframes = []

for i, text in enumerate(texts, start=1):
    doc = nlp(text)

    entities = []
    labels = []
    position_start = []
    position_end = []

    for ent in doc.ents:
        entities.append(ent.text)
        labels.append(ent.label_)
        position_start.append(ent.start_char)
        position_end.append(ent.end_char)

    df = pd.DataFrame({'Entities': entities, 'Labels': labels, 'Position_Start': position_start, 'Position_End': position_end})
    all_dataframes.append(df)

    # we print the dataframe for each document
    print(f"DataFrame for Document {i}:")
    print(df)
    print("\n")


# Now, all_dataframes is a list containing the dataframes for each document
# You can access them using all_dataframes[0], all_dataframes[1], and all_dataframes[2]
```

```
DataFrame for Document 1:
            Entities    Labels  Position_Start  Position_End
0               half  CARDINAL             186           190
1          two years      DATE             418           427
2                Tom    PERSON             458           461
3       Mrs\nParsons    PERSON             480           491
4            Winston    PERSON             535           542
5   the Youth League       ORG             944           960
6              Spies       ORG             969           974
7                one  CARDINAL            1147          1150
8              first   ORDINAL            1168          1173
```

| | | | | |
|---|---|---|---|---|
| 9 | telescreen | CARDINAL | 1412 | 1422 |
| 10 | Mrs Parsons | PERSON | 1450 | 1461 |
| 11 | half | CARDINAL | 1473 | 1477 |
| 12 | today | DATE | 1534 | 1539 |

DataFrame for Document 2:

| | Entities | Labels | Position_Start | Position_End |
|---|---|---|---|---|
| 0 | colonies | GPE | 248 | 256 |
| 1 | The Ministry of Truth | ORG | 471 | 492 |
| 2 | Newspeak | GPE | 506 | 514 |
| 3 | Oceania | LOC | 554 | 561 |
| 4 | 300 metres | QUANTITY | 786 | 796 |
| 5 | Winston | PERSON | 822 | 829 |
| 6 | three | CARDINAL | 921 | 926 |
| 7 | Party | ORG | 942 | 947 |
| 8 | IGNORANCE | ORG | 981 | 990 |
| 9 | The Ministry of Truth | ORG | 1003 | 1024 |
| 10 | three | CARDINAL | 1049 | 1054 |
| 11 | thousand | CARDINAL | 1055 | 1063 |
| 12 | London | GPE | 1145 | 1151 |
| 13 | three | CARDINAL | 1168 | 1173 |

DataFrame for Document 3:

| | Entities | Labels | Position_Start | Position_End |
|---|---|---|---|---|
| 0 | four | CARDINAL | 4 | 8 |
| 1 | four | CARDINAL | 60 | 64 |
| 2 | The Ministry of Truth | ORG | 138 | 159 |
| 3 | The Ministry of Peace | ORG | 240 | 261 |
| 4 | The Ministry of Love | ORG | 296 | 316 |
| 5 | the Ministry of Plenty | ORG | 354 | 376 |
| 6 | Newspeak | LANGUAGE | 438 | 446 |
| 7 | Minipax | NORP | 458 | 465 |
| 8 | Miniluv | GPE | 467 | 474 |
| 9 | Miniplenty | PERSON | 480 | 490 |
| 10 | The Ministry of Love | ORG | 492 | 512 |
| 11 | Winston | PERSON | 581 | 588 |
| 12 | the Ministry of Love | ORG | 611 | 631 |
| 13 | half a kilometre | QUANTITY | 644 | 660 |
| 14 | Winston | PERSON | 984 | 991 |
| 15 | tomorrow | DATE | 1389 | 1397 |
| 16 | Chinese | NORP | 1554 | 1561 |
| 17 | Winston | PERSON | 1575 | 1582 |

```
manual_labels1 = [("Tom", "PERSON"), ("Mrs Parsons", "PERSON"),("Winston","PERSON"), ("the Youth League", "ORG"), ("today", "DATE")]

manual_labels2 = [("Tom", "PERSON"), ("The Ministry of Truth", "ORG"), ("Newspeak", "GPE")]

manual_labels3 = [("Tom", "PERSON"), ("the Ministry of Love", "ORG"), ("tomorrow", "DATE"),("four","CARDINAL")]
```

```python
# Calculate precision, recall, and F1 score
def calculate_metrics(predicted, actual):
    true_positives = len(set(predicted) & set(actual))
    false_positives = len(set(predicted) - set(actual))
    false_negatives = len(set(actual) - set(predicted))

    precision = true_positives / (true_positives + false_positives + 1e-9)if (true_positives + false_positives) != 0 else 0
    recall = true_positives / (true_positives + false_negatives + 1e-9)if (true_positives + false_positives) != 0 else 0
    f1_score = 2 * (precision * recall) / (precision + recall + 1e-9)if (precision + recall) != 0 else 0

    return precision, recall, f1_score
```

```python
doc1 = nlp(text1)
doc2 = nlp(text2)
doc3 = nlp(text3)
entities1 = [(ent.text, ent.label_) for ent in doc1.ents]
entities2 = [(ent.text, ent.label_) for ent in doc2.ents]
entities3 = [(ent.text, ent.label_) for ent in doc3.ents]



# Calculate metrics
precision, recall, f1_score = calculate_metrics(entities1,manual_labels1)
df1 = pd.DataFrame({'Text': [1], 'Precision': precision, 'Recall': recall, 'F1 Score': f1_score})
precision, recall, f1_score = calculate_metrics(entities2,manual_labels2)
df2 = pd.DataFrame({'Text': [2], 'Precision': precision, 'Recall': recall, 'F1 Score': f1_score})
precision, recall, f1_score = calculate_metrics(entities3,manual_labels3)
df3 = pd.DataFrame({'Text': [3], 'Precision': precision, 'Recall': recall, 'F1 Score': f1_score})


# List of DataFrames
dfs = [df1, df2, df3]

# Combine DataFrames vertically
rdf = pd.concat(dfs, ignore_index=True)

# Print metrics
rdf
```

| | Text | Precision | Recall | F1 Score |
|---|---|---|---|---|
| **0** | 1 | 0.416667 | 1.000000 | 0.588235 |
| **1** | 2 | 0.181818 | 0.666667 | 0.285714 |
| **2** | 3 | 0.214286 | 0.750000 | 0.333333 |

```
textall = []

my_string_array = ["one", "two", "three", "four", "five", "six", "seven", "eight", "nine", "ten",
                   "eleven", "twelve"]

for i in range(1, 12):
    chapter_number = my_string_array[i]

    if chapter_number in tokens:
        start_index = tokens.index(chapter_number) + 1

        if i < 11:
            end_index = tokens.index(my_string_array[i+1], start_index)
        else:
            # For the last chapter, use the end of the list
            end_index = len(tokens)

        chapter_content = tokens[start_index:end_index]
        chapter_text = ' '.join(chapter_content)
        textall.append(chapter_text)


        # Your code using the 'chapter_content' variable goes here
        print(f"Chapter {i} content:", chapter_text)
    else:
        print(f"Chapter {i} not found in the 'tokens' list.")
```

```
Chapter 1 content: minuteshate room 101 memory hole newspeak doublethink prolesunperson and thoughtcrimepart one 1984chapter 1it was a bright cold day in ap
Chapter 2 content: other buildings of similar appearance and sizeso completely did they dwarf the surrounding architecture that from the roof of victory ma
Chapter 3 content: of them simultaneously they were the homes ofthe four ministries between which the entire apparatusof government was divided the ministr
Chapter 4 content: minutes and it waspossible that his features had not been perfectly under control it was terribly dangerous to let your thoughts wanderw
Chapter 5 content: steps down the passage when somethinghit the back of his neck an agonizingly painful blow it wasas though a redhot wire had been jabbed
Chapter 6 content: flights upand winston who was thirtynine and had a varicose ulcerabove his right ankle went slowly resting several times onthe way on ea
Chapter 7 content: years earlierthe story really began in the middle sixties the period ofthe great purges in which the original leaders of the revolution
Chapter 8 content: had popped upfrom behind the table and was menacing him with a toyautomatic pistol while his small sister about two yearsyounger made th
Chapter 9 content: minutes he had to be back at work by fourteenthirtycuriously the chiming of the hour seemed to have putnew heart into him he was a lonely
Chapter 10 content: hundred and in the records department where winston worked they were dragging thechairs out of the cubicles and grouping them in the ce
Chapter 11 content: they passed through a brief blossomingperiod of beauty and sexual desire they married attwenty they were middleaged at thirty they died
```

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
#let's create the vectorizer and fit the corpus and transform them accordingly
v = TfidfVectorizer()

v.fit(textall)
transform_output = v.transform(textall)
```

```
#let's print the vocabulary

print(v.vocabulary_)
```


{'minuteshate': 8457, 'room': 11228, '101': 1, 'memory': 8301, 'hole': 6509, 'newspeak': 8831, 'doublethink': 3835, 'prolesunperson': 10490, 'and': 621, 'thoughtcrimepart': 13968, 'one': 9373, '1984chapter': 31, '1it': 36, 'was': 15059, 'bright': 1929, 'cold': 2619, 'day': 3279, 'in': 6815, 'april': 936, 'the': 13331, 'clocks': 2555, 'were': 15297, 'striking': 12745, 'thirteen': 13908, 'winston': 15600, 'smith': 12149,

```
'his': 6421, 'chin': 2433, 'nuzzled': 9069, 'into': 7130, 'hisbreast':
6426, 'an': 606, 'effort': 4279, 'to': 14126, 'escape': 4498, 'vile':
14917, 'wind': 15582, 'slipped': 12094, 'quicklythrough': 10654,
'glass': 5647, 'doors': 3820, 'of': 9145, 'victory': 14909, 'mansions':
8125, 'though': 13956, 'notquickly': 9010, 'enough': 4426, 'prevent':
10395, 'swirl': 13035, 'gritty': 5821, 'dust': 3977, 'from': 5426,
'entering': 4447, 'along': 517, 'with': 15671, 'himthe': 6403,
'hallway': 5997, 'smelt': 12145, 'boiled': 1783, 'cabbage': 2113,
'old': 9329, 'rag': 10685, 'mats': 8205, 'atone': 1146, 'end': 4371,
'it': 7261, 'col
```

```
v.get_feature_names_out()
```

```
array(['100', '101', '101and', ..., 'zip', 'zipper', 'zoom'],
dtype=object)
```

```
#Focus on IDF VALUES
print(v.idf_)
```

```
[1.87546874 2.38629436 2.79175947 ... 2.79175947 2.79175947 2.79175947]
```

```
#let's print the idf of each word:

all_feature_names = v.get_feature_names_out()

for word in all_feature_names:

    #let's get the index in the vocabulary
    indx = v.vocabulary_.get(word)

    #get the score
    idf_score = v.idf_[indx]

    print(f"{word} : {idf_score}")
```

**Streaming output truncated to the last 5000 lines.**

```
rest : 1.538996500732687
restand : 2.791759469228055
rested : 2.791759469228055
```

```
resting : 2.09861228866811
restless : 1.8754687373538999
restore : 2.791759469228055
restored : 2.09861228866811
restoring : 2.09861228866811
restricted : 2.791759469228055
restricting : 2.791759469228055
rests : 2.791759469228055
result : 2.791759469228055
resultingamalgam : 2.791759469228055
resultof : 2.791759469228055
results : 2.791759469228055
resultwhatever : 2.791759469228055
resurrected : 2.386294361119891
resurrection : 2.791759469228055
retained : 1.8754687373538999
```

```python
#let's print the transformed output from tf-idf
q=transform_output.toarray()

# Create a DataFrame to display the TF-IDF matrix
tfidf_matrix = pd.DataFrame(q, columns=all_feature_names, index=['Chapter {}'.format(i+1) for i in range(len(textall))])


# Set NumPy print options
np.set_printoptions(threshold=np.inf, linewidth=np.inf)

# Display the entire array
tfidf_matrix
```

| | 100 | 101 | 101and | 101he | 101room | 101the | 1261984suddenly | 14284 | 145 | 15 | ... | youwill | youwinston | youwould | yp | zeal | z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Chapter 1 | 0.000000 | 0.022268 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0. |
| Chapter 2 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0. |
| Chapter 3 | 0.001117 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.001250 | 0.002500 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.001250 | 0.002500 | 0 |
| Chapter 4 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0. |
| Chapter 5 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0. |
| Chapter 6 | 0.000877 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000981 | 0.001963 | 0.001116 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000981 | 0.001963 | 0. |
| Chapter 7 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.005245 | 0.000000 | 0.000000 | 0. |
| Chapter 8 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0. |
| Chapter 9 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0. |
| Chapter 10 | 0.001052 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.001178 | 0.002355 | 0.000000 | ... | 0.000000 | 0.000000 | 0.000000 | 0.001178 | 0.002355 | 0. |
| Chapter 11 | 0.000547 | 0.003133 | 0.000407 | 0.000407 | 0.000815 | 0.000407 | 0.000407 | 0.000000 | 0.000000 | 0.000348 | ... | 0.002444 | 0.000815 | 0.000348 | 0.000000 | 0.000000 | 0. |

```python
from sklearn.metrics.pairwise import cosine_similarity
import seaborn as sns
# Function to calculate cosine similarity between TF-IDF vectors
def calculate_cosine_similarity(tfidf_matrix):
    similarity_matrix = cosine_similarity(tfidf_matrix, tfidf_matrix)
    return similarity_matrix

# Function to visualize the similarity matrix
def visualize_similarity_matrix(similarity_matrix, chapter_names):
    plt.figure(figsize=(200, 150))
    sns.heatmap(similarity_matrix, annot=True, fmt=".2f", xticklabels=chapter_names, yticklabels=chapter_names, cmap="viridis")
    plt.title("Chapter Similarity")
    plt.show()

chapter_names = [f"Chapter {i}" for i in range(1, 40)]

    # Calculate cosine similarity
similarity_matrix = calculate_cosine_similarity(tfidf_matrix)

# Visualize the similarity matrix

visualize_similarity_matrix(similarity_matrix, chapter_names)
```

```python
# Create a DataFrame to display the similarity score matrix
similarity_score = pd.DataFrame(similarity_matrix, columns=['Chapter {}'.format(i+1) for i in range(len(textall))], index=['Chapter {}'.format(i+1) for i in range(len(textall))])


# Set NumPy print options
np.set_printoptions(threshold=np.inf, linewidth=np.inf)

# Display the entire array
similarity_score
```

| | Chapter 1 | Chapter 2 | Chapter 3 | Chapter 4 | Chapter 5 | Chapter 6 | Chapter 7 | Chapter 8 | Chapter 9 | Chapter 10 | Chapter 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Chapter 1** | 1.000000 | 0.322037 | 0.858628 | 0.783642 | 0.761258 | 0.868430 | 0.835578 | 0.842409 | 0.707819 | 0.857000 | 0.846730 |
| **Chapter 2** | 0.322037 | 1.000000 | 0.323200 | 0.279571 | 0.295680 | 0.326768 | 0.319780 | 0.327230 | 0.248233 | 0.323525 | 0.330384 |
| **Chapter 3** | 0.858628 | 0.323200 | 1.000000 | 0.922969 | 0.847001 | 0.997651 | 0.953648 | 0.939603 | 0.837390 | 0.997779 | 0.976928 |
| **Chapter 4** | 0.783642 | 0.279571 | 0.922969 | 1.000000 | 0.772705 | 0.941639 | 0.898138 | 0.858637 | 0.788829 | 0.941875 | 0.928248 |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **Chapter 5** | 0.761258 | 0.295680 | 0.847001 | 0.772705 | 1.000000 | 0.846921 | 0.813270 | 0.890391 | 0.723232 | 0.844926 | 0.821342 |
| **Chapter 6** | 0.868430 | 0.326768 | 0.997651 | 0.941639 | 0.846921 | 1.000000 | 0.956847 | 0.939264 | 0.836674 | 0.998647 | 0.980273 |
| **Chapter 7** | 0.835578 | 0.319780 | 0.953648 | 0.898138 | 0.813270 | 0.956847 | 1.000000 | 0.906518 | 0.814056 | 0.954273 | 0.956319 |
| **Chapter 8** | 0.842409 | 0.327230 | 0.939603 | 0.858637 | 0.890391 | 0.939264 | 0.906518 | 1.000000 | 0.795841 | 0.937905 | 0.926109 |
| **Chapter 9** | 0.707819 | 0.248233 | 0.837390 | 0.788829 | 0.723232 | 0.836674 | 0.814056 | 0.795841 | 1.000000 | 0.835310 | 0.830552 |
| **Chapter 10** | 0.857000 | 0.323525 | 0.997779 | 0.941875 | 0.844926 | 0.998647 | 0.954273 | 0.937905 | 0.835310 | 1.000000 | 0.979318 |
| **Chapter 11** | 0.846730 | 0.330384 | 0.976928 | 0.928248 | 0.821342 | 0.980273 | 0.956319 | 0.926109 | 0.830552 | 0.979318 | 1.000000 |