```python
import nltk
import requests
from bs4 import BeautifulSoup
import matplotlib.pyplot as plt

# Download necessary NLTK datasets
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('omw-1.4')
nltk.download('averaged_perceptron_tagger')
nltk.download('averaged_perceptron_tagger_eng')
nltk.download('maxent_ne_chunker')
nltk.download('maxent_ne_chunker_tab')
nltk.download('words')
nltk.download('punkt_tab')

print("Setup Complete. Libraries Loaded.")
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package averaged_perceptron_tagger_eng to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger_eng is already up-t
[nltk_data]       date!
[nltk_data] Downloading package maxent_ne_chunker to
[nltk_data]     /root/nltk_data...
[nltk_data]   Package maxent_ne_chunker is already up-to-date!
[nltk_data] Downloading package maxent_ne_chunker_tab to
[nltk_data]     /root/nltk_data...
Setup Complete. Libraries Loaded.
[nltk_data]   Unzipping chunkers/maxent_ne_chunker_tab.zip.
[nltk_data] Downloading package words to /root/nltk_data...
[nltk_data]   Package words is already up-to-date!
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
```

```
#URL to be scraped
url = "https://en.wikipedia.org/wiki/Lionel_Messi"

# Defining headers to mimic a real browser (This tells the server:
headers = {
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleW
}

#Request to fetch the webpage content WITH HEADERS
response = requests.get(url, headers=headers)

#Check if the request was successful (Status Code 200)
if response.status_code == 200:
    print("Successfully fetched the webpage!")

    #Parse the HTML content
    soup = BeautifulSoup(response.content, 'html.parser')

    #Extract text from paragraph <p> tags only
    paragraphs = soup.find_all('p')
    text_data = " ".join([p.get_text() for p in paragraphs])
    print("\n--- Scraped Text Preview ---")
    print(text_data[:500])

else:
    print(f"Failed to retrieve the webpage. Status Code: {response.
    text_data = ""
```

```
Successfully fetched the webpage!

--- Scraped Text Preview ---


 Lionel Andrés "Leo" Messi[note 1] (born 24 June 1987) is an Argenti
```

```
from nltk.tokenize import sent_tokenize, word_tokenize

# Tokenize sentences
sentences = sent_tokenize(text_data)
print("Sentence Count:", len(sentences))

# Tokenize words
words = word_tokenize(text_data)
print("Word Count:", len(words))
```

```
Sentence Count: 447
Word Count: 14267
```

```
from nltk.corpus import stopwords
import string

stop_words = set(stopwords.words("english"))

# Remove stop words and punctuation
filtered_words = [w for w in words if w.lower() not in stop_words a

print("Filtered Word Count:", len(filtered_words))
```

```
Filtered Word Count: 7512
```

```
from nltk.stem import PorterStemmer, WordNetLemmatizer

ps = PorterStemmer()
lemmatizer = WordNetLemmatizer()

print(f"{'Word':<15} | {'Stemmed':<15} | {'Lemmatized':<15}")
print("-" * 50)

# Compare first 10 words
for w in filtered_words[:10]:
    print(f"{w:<15} | {ps.stem(w):<15} | {lemmatizer.lemmatize(w, p
```

```
Word            | Stemmed         | Lemmatized
--------------------------------------------------
Lionel          | lionel          | Lionel
Andrés          | andré           | Andrés
``              | ``              | ``
Leo             | leo             | Leo
''              | ''              | ''
Messi           | messi           | Messi
note            | note            | note
1               | 1               | 1
born            | born            | bear
24              | 24              | 24
```

```
from nltk import pos_tag, ne_chunk

# POS Tagging
pos_tags = pos_tag(filtered_words)
print("POS Tags Sample:", pos_tags[:5])

# Named Entity Recognition
ner_tree = ne_chunk(pos_tag(words))

print("\n--- Entities Detected ---")
for chunk in ner_tree:
    if hasattr(chunk, 'label'):
        print(chunk.label(), ' '.join(c[0] for c in chunk))
```

```
POS Tags Sample: [('Lionel', 'NNP'), ('Andrés', 'NNP'), ('``', '``

--- Entities Detected ---
PERSON Lionel
PERSON Andrés
PERSON Messi
PERSON League Soccer
ORGANIZATION Inter Miami
GPE Argentina
PERSON Messi
```

```
GPE Ballon
ORGANIZATION European Golden Shoes
ORGANIZATION FIFA
ORGANIZATION All Time Men
ORGANIZATION IFFHS
PERSON Messi
GPE Barcelona
GPE La Liga
ORGANIZATION FIFA
ORGANIZATION Copa América
PERSON Messi
PERSON Messi
PERSON Barcelona
PERSON Barcelona
GPE Spanish
GPE Messi
GPE Ballon
GPE European
PERSON Barcelona
GPE La Liga
PERSON Barcelona
GPE Ballon
PERSON Barcelona
PERSON Ballon
GPE Barcelona
PERSON Messi
ORGANIZATION UEFA
ORGANIZATION Barcelona
PERSON Messi
GPE French
PERSON Paris
ORGANIZATION MLS
ORGANIZATION Inter Miami
ORGANIZATION MLS
ORGANIZATION MVP
ORGANIZATION Argentine
PERSON Messi
ORGANIZATION Summer Olympics
GPE Argentina
ORGANIZATION Copa América Centenario
PERSON Messi
GPE Argentina
GPE Ballon
GPE Argentina
GPE Ballon
PERSON Messi
PERSON Messi
PERSON Adidas
```

```
from nltk.corpus import wordnet

# Lookup definition
term = "football"
syns = wordnet.synsets(term)

if syns:
    print(f"Def ({term}):", syns[0].definition())
```

```
Def (football): any of various games played with a ball (round or ov
```

ॐ नमः शिवाय

NLP Exp : 02 Date :

Post Lab Questions

A1 In Python (specifically NLTK), a corpus is used as a large, structured collection of text data (like movie reviews etc.) to train NLP models, perform statistical analysis & test algorithms for patterns, sentiment or frequency

A2 A good corpus is large, structured, representative and often annotated

A3 Tokenization, Stop Word Removal, Stemming, Lemmatization, Part of Speech (~~POS~~ (POS) Tagging, Frequency Distribution etc.

Conclusion :

In this experiment, we successfully analyzed unstructured text about Lionel Messi using Python & NLTK. Techniques like Tokenization and Stop - Word Removal filtered out noise, while Frequency Distribution identified core themes.

The O/P correctly highlights "Messi" & "Football" as top keywords proving the code effectively extracts insights from raw data.

Rainbow