Assignment 3

Q1) Define Natural Language Processing (NLP). Provide three real-world applications of NLP and explain how they impact society.

```
Name: Sidhanta Barik, RegNo: 2241002049
```

Ans: Natural Language Processing (NLP) is a field of artificial intelligence that focuses on the interaction between computers and humans through natural language. The goal of NLP is to enable computers to understand, interpret, and generate human language in a way that is both meaningful and useful.

3 real-world applications of NLP and their impact on society:

- **Chatbots and Virtual Assistants**: NLP is used to develop chatbots and virtual assistants like Siri, Alexa, and Google Assistant. These tools help users perform tasks, answer questions, and provide customer support, making daily life more convenient and improving customer service efficiency.
- **Language Translation**: NLP powers translation services such as Google Translate, allowing users to translate text and speech between different languages. This breaks down language barriers, facilitates global communication, and promotes cultural exchange.
- **Sentiment Analysis**: NLP is used to analyze social media posts, reviews, and feedback to determine the sentiment behind the text. Businesses use sentiment analysis to understand customer opinions, improve products and services, and make data-driven decisions, ultimately enhancing customer satisfaction and loyalty.

Q2) Explain the following terms and their significance in NLP:

- Tokenization
- Stemming
- Lemmatization

```
  Name: Sidhanta Barik, RegNo: 2241002049
```

Ans:-

- **Tokenization**: Tokenization is the process of breaking down text into smaller units called tokens, which can be words, phrases, or sentences. It is a fundamental step in NLP as it helps in understanding the structure and meaning of the text. Tokenization is crucial for tasks such as text analysis, information retrieval, and machine learning.

- **Stemming**: Stemming is the process of reducing words to their base or root form. It involves removing suffixes and prefixes to obtain the stem of a word. Stemming helps in normalizing text, reducing the complexity of data, and improving the performance of NLP tasks like text classification and information retrieval.
- **Lemmatization**: Lemmatization is the process of reducing words to their base or dictionary form, known as the lemma. Unlike stemming, lemmatization considers the context and morphological analysis of words, ensuring that the root form is a valid word. Lemmatization enhances the accuracy of NLP tasks by providing meaningful and contextually appropriate root forms of words.

Q3) What is Part-of-Speech (POS) tagging? Discuss its importance with an example.

```
Name: Sidhanta Barik, RegNo: 2241002049
```

Ans: Part-of-Speech (POS) tagging is the process of assigning a part of speech to each word in a given text, such as nouns, verbs, adjectives, adverbs, etc. POS tagging is essential in NLP as it helps in understanding the grammatical structure of a sentence, which is crucial for various NLP tasks such as parsing, named entity recognition, and machine translation.

Importance of POS tagging:

- **Disambiguation**: POS tagging helps in resolving ambiguities in words that can have multiple meanings based on their usage in a sentence. For example, the word "book" can be a noun (a physical object) or a verb (to reserve).
- **Syntactic Parsing**: POS tags provide the syntactic structure of a sentence, which is essential for parsing and understanding the relationships between words.
- **Information Retrieval**: POS tagging improves the accuracy of information retrieval systems by enabling more precise searches based on the grammatical roles of words.

Example: Consider the sentence "The quick brown fox jumps over the lazy dog."

- The: Determiner (DT)
- quick: Adjective (JJ)
- brown: Adjective (JJ)
- fox: Noun (NN)
- jumps: Verb (VBZ)
- over: Preposition (IN)
- the: Determiner (DT)
- lazy: Adjective (JJ)
- dog: Noun (NN)

In this example, POS tagging helps identify the grammatical roles of each word, which aids in understanding the sentence structure and meaning.

Q4) Create a TextBlob named exercise blob containing ”This is a TextBlob”.

```
print('''Name: Sidhanta Barik, RegNo: 2241002049
-----------------------------------------------------''')

from textblob import TextBlob
exercise_blob = TextBlob("This is a TextBlob")
print(exercise_blob)
```

```
Name: Sidhanta Barik, RegNo: 2241002049
-----------------------------------------------------

This is a TextBlob
```

Q5) Write a Python script to perform the following tasks on the given text:

- Tokenize the text into words and sentences.
- Perform stemming and lemmatization using NLTK or SpaCy.
- Remove stop words from the text.
- Sample Text: ”Natural Language Processing enables machines to understand and process human languages. It is a fascinating field with numerous applications, such as chatbots and language translation.”

```
print('''Name: Sidhanta Barik, RegNo: 2241002049
-----------------------------------------------------''')

import spacy
nlp = spacy.load('en_core_web_sm')
from textblob import TextBlob, Word
text = "Natural Language Processing enables machines to understand and
process human languages. It is a fascinating field with numerous
applications, such as chatbots and language translation."
blob = TextBlob(text)
sentList = [s for s in blob.sentences]
print(f"Sentence Tokenization:-\n{sentList}")
wordList = [w for w in blob.words]
print(f"\nWord Tokenization:-\n{wordList}")
print("\nStemming:-")
word = Word("running")
print(word.stem())
print("\nLemmatization:-")
print(word.lemmatize())
print("\nAfter Removing Stop Words:-")
for word in blob.words:
    if word not in nlp.Defaults.stop_words:
        print(word, end=" ")
```

```
Name: Sidhanta Barik, RegNo: 2241002049
-----------------------------------------------------
Sentence Tokenization:-
[Sentence("Natural Language Processing enables machines to understand
and process human languages."), Sentence("It is a fascinating field
with numerous applications, such as chatbots and language
translation.")]

Word Tokenization:-
['Natural', 'Language', 'Processing', 'enables', 'machines', 'to',
'understand', 'and', 'process', 'human', 'languages', 'It', 'is', 'a',
'fascinating', 'field', 'with', 'numerous', 'applications', 'such',
'as', 'chatbots', 'and', 'language', 'translation']

Stemming:-
run

Lemmatization:-
running

After Removing Stop Words:-
Natural Language Processing enables machines understand process human
languages It fascinating field numerous applications chatbots language
translation
```

Q6) Web Scraping with the Requests and Beautiful Soup Libraries:

- Use the requests library to download the www.python.org home page's content.
- Use the Beautiful Soup library to extract only the text from the page.
- Eliminate the stop words in the resulting text, then use the wordcloud module to create a word cloud based on the text.

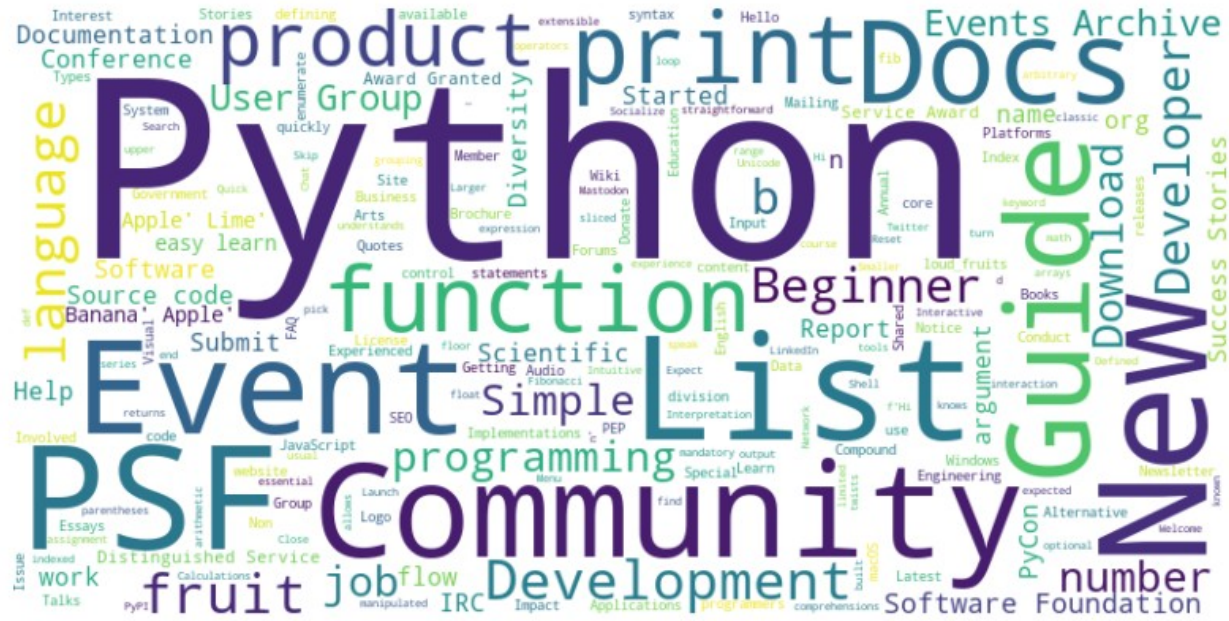
```python
print('''Name: Sidhanta Barik, RegNo: 2241002049
-----------------------------------------------------''')

import requests
from bs4 import BeautifulSoup # Install bs4
from wordcloud import WordCloud
import matplotlib.pyplot as plt

url = "https://www.python.org"
response = requests.get(url)
html_content = response.text
soup = BeautifulSoup(html_content, 'html.parser')
page_text = soup.get_text()
filtered_words = []
for word in page_text.split():
    if word.lower() not in nlp.Defaults.stop_words:
```

```
        filtered_words.append(word)
wordcloud = WordCloud(width=800, height=400,
background_color='white').generate(' '.join(filtered_words))
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()

Name: Sidhanta Barik, RegNo: 2241002049
-----------------------------------------------------
```



Q7) (Tokenizing Text and Noun Phrases) Using the text from above problem, create a TextBlob, then tokenize it into Sentences and Words, and extract its noun phrases.

```
print('''Name: Sidhanta Barik, RegNo: 2241002049
-----------------------------------------------------''')
from textblob import TextBlob

text = "Natural Language Processing enables machines to understand and
process human languages. It is a fascinating field with numerous
applications, such as chatbots and language translation."
blob = TextBlob(text)
print(f"Sentences: {blob.sentences}")
print(f"\nWords: {blob.words}")
print(f"\nNoun Phrases: {blob.noun_phrases}")

Name: Sidhanta Barik, RegNo: 2241002049
-----------------------------------------------------
```

```
Sentences: [Sentence("Natural Language Processing enables machines to
understand and process human languages."), Sentence("It is a
fascinating field with numerous applications, such as chatbots and
language translation.")]

Words: ['Natural', 'Language', 'Processing', 'enables', 'machines',
'to', 'understand', 'and', 'process', 'human', 'languages', 'It',
'is', 'a', 'fascinating', 'field', 'with', 'numerous', 'applications',
'such', 'as', 'chatbots', 'and', 'language', 'translation']

Noun Phrases: ['language processing', 'process human languages',
'numerous applications', 'language translation']
```

Q8) (Sentiment of a News Article) Using the techniques in problem no. 5, download a web page for a current news article and create a TextBlob. Display the sentiment for the entire TextBlob and for each Sentence.

```
print('''Name: Sidhanta Barik, RegNo: 2241002049
-----------------------------------------------------''')
import requests
from bs4 import BeautifulSoup
from textblob import TextBlob
url = "https://www.bbc.com/news"
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser')
text = soup.get_text()
blob = TextBlob(text)
sentences = blob.sentences[6:8]
print(f"Overall Sentiment: {blob.sentiment}\n")
for s in sentences:
    print(f"Sentence: {s}")
    print(f"Sentiment: {s.sentiment}")
    print()
```

```
Name: Sidhanta Barik, RegNo: 2241002049
-----------------------------------------------------
Overall Sentiment: Sentiment(polarity=0.0047149122807017664,
subjectivity=0.3721491228070174)

Sentence: 1 during Hollywood strikes.
Sentiment: Sentiment(polarity=0.0, subjectivity=0.0)

Sentence: Now it's in Oscar-winning filmsTech companies say AI will
make productions cheaper and faster - but many in Hollywood think it
will replace their jobs.8 hrs agoUS & CanadaTrump wants India to buy
US corn - but here's why it probably won't India feeds over a billion
people, yet low yields and poor infrastructure keep agriculture
```

```
lagging.14 hrs agoWorldRichard Chamberlain: Heartthrob king of the TV
mini-seriesDashing American actor who was unrivalled in his ability to
hold a television audience.23 hrs agoCultureMajor earthquake adds to
war-torn Myanmar's troublesThe quake comes at a time of ongoing civil
war, food shortages and a declining economy.
Sentiment: Sentiment(polarity=0.059999999999999984, subjectivity=0.3)
```

Q9) (Sentiment of a News Article with the NaiveBayesAnalyzer) Repeat the previous exercise but use the NaiveBayesAnalyzer for sentiment analysis.

```
print('''Name: Sidhanta Barik, RegNo: 2241002049
-------------------------------------------------------''')
import requests
from bs4 import BeautifulSoup
from textblob import TextBlob
from textblob.sentiments import NaiveBayesAnalyzer
url = "https://www.bbc.com/news"
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser')
text = soup.get_text()
blob = TextBlob(text, analyzer=NaiveBayesAnalyzer())
sentences = blob.sentences[6:8]
print(f"Overall Sentiment: {blob.sentiment}\n")
for s in sentences:
    print(f"Sentence: {s}")
    print(f"Sentiment: {s.sentiment}")
    print()
```

```
Name: Sidhanta Barik, RegNo: 2241002049
-------------------------------------------------------
Overall Sentiment: Sentiment(classification='pos', p_pos=1.0,
p_neg=3.140062957303287e-16)

Sentence: 1 during Hollywood strikes.
Sentiment: Sentiment(classification='pos', p_pos=0.6903883969389157,
p_neg=0.30961160306108393)

Sentence: Now it's in Oscar-winning filmsTech companies say AI will
make productions cheaper and faster - but many in Hollywood think it
will replace their jobs.9 hrs agoUS & CanadaTrump wants India to buy
US corn - but here's why it probably won't India feeds over a billion
people, yet low yields and poor infrastructure keep agriculture
lagging.14 hrs agoWorldRichard Chamberlain: Heartthrob king of the TV
mini-seriesDashing American actor who was unrivalled in his ability to
hold a television audience.23 hrs agoCultureMajor earthquake adds to
war-torn Myanmar's troublesThe quake comes at a time of ongoing civil
war, food shortages and a declining economy.
```

```
Sentiment: Sentiment(classification='pos', p_pos=0.9469750795096648,
p_neg=0.053024920490335185)
```

Q10) (Spell Check a Project Gutenberg Book) Download a Project Gutenberg book and create a TextBlob. Tokenize the TextBlob into Words and determine whether any are misspelled. If so, display the possible corrections.

```
print('''Name: Sidhanta Barik, RegNo: 2241002049
-------------------------------------------------''')
import requests
from textblob import TextBlob
url = "https://www.gutenberg.org/cache/epub/64317/pg64317.txt"
text = requests.get(url).text[3000:4000]
blob = TextBlob(text)
for word in blob.words:
    if word.correct() != word:
        print(f"Misspelled Word: {word}, Corrected: {word.correct()}")
```

```
Name: Sidhanta Barik, RegNo: 2241002049
-------------------------------------------------
Misspelled Word: tolerance, Corrected: tolerable
Misspelled Word: East, Corrected: Last
Misspelled Word: If, Corrected: Of
Misspelled Word: This, Corrected: His
Misspelled Word: —it, Corrected: it
```

Q11)

- Write a Python program that takes user input in English and translates it to French, Spanish, and German using TextBlob.
- Create a program that takes multiple user-inputted sentences, analyzes polarity and subjectivity, and categorizes them as objective/subjective and positive/negative/neutral.
- Develop a function that takes a paragraph, splits it into sentences, and calculates the sentiment score for each sentence individually.
- Write a program that takes a sentence as input and prints each word along with its POS tag using TextBlob.
- Create a function that takes a user-inputted word, checks its spelling using TextBlob, and suggests top 3 closest words if a mistake is found.
- Build a Python script that extracts all adjectives from a given para and prints them in order of occurrence.
- Write a program that takes a news article as input and extracts the top 5 most common noun phrases as keywords.
- Write a program that takes a news article as input and extracts the top 5 most common noun phrases as keywords.

- Write a program that summarizes a given para by keeping only the most informative sentences, based on noun phrase frequency.

```
print('''Name: Sidhanta Barik, RegNo: 2241002049
-------------------------------------------------''')
# Translation with textblob does not work.

from deep_translator import GoogleTranslator as gt
# text = input("Enter something to translate: ")
text = "Hello World"
print(f"Original Text: {text}")
print(f"French: {gt(source="auto", target="fr").translate(text)}")
print(f"Spanish: {gt(source="auto", target="es").translate(text)}")
print(f"German: {gt(source="auto", target="de").translate(text)}")
print()

from textblob import TextBlob
# para = input("Enter a paragraph: ")
para = "Greetings World. I am Sid. I am happy to learn Python. I am
sad today."
print(f"Original Paragraph: {para}")
blob = TextBlob(para)
sents = blob.sentences
for i, s in enumerate(sents):
    print(f"Sentence {i+1}: {s}")
    print(f"Sentiment: {s.sentiment}")
    print()

# sent1 = input("Enter a sentence: ")
sent1 = "Love never came to me in a silver spoon, so i licked it off
knives."
blob = TextBlob(sent1)
print(f"POS Tagging: {blob.tags}")
print()

# w = input("Enter a word: ")
w = "Theyr"
blob = TextBlob(w)
correctedW = str(blob.correct())
if correctedW == w:
    print(f"The word '{w}' is spelled correctly.")
else:
    suggestions = blob.words[0].spellcheck()[:3]
    suggestion_list = [s[0] for s in suggestions]
    print(f"The word '{w}' is misspelled. Top 3 suggestions are: {',
'.join(suggestion_list)}")
print()

import nltk
from nltk.tokenize import word_tokenize
words = word_tokenize(para)
```

```
adjectives = []
for word in words:
    tags = nltk.pos_tag([word])
    if tags and tags[0][1].startswith('JJ'):
        adjectives.append(word)
print(f"Paragraph: {para}")
print("Adjectives in order of occurrence:")
for adj in adjectives:
    print(adj)
print()

from collections import Counter
# para = input("Enter a paragraph: ")
sentences = sent_tokenize(para)
noun_phrases = []
for sentence in sentences:
    words = word_tokenize(sentence)
    tagged_words = nltk.pos_tag(words)
    temp_phrase = []
    for word, tag in tagged_words:
        if tag.startswith('NN'):
            temp_phrase.append(word.lower())
        elif temp_phrase:
            noun_phrases.append(" ".join(temp_phrase))
            temp_phrase = []
    if temp_phrase:
        noun_phrases.append(" ".join(temp_phrase))
noun_phrase_counts = Counter(noun_phrases)
sentence_scores = {}
for i, sentence in enumerate(sentences):
    score = 0
    words_in_sentence = word_tokenize(sentence.lower())
    for phrase, count in noun_phrase_counts.items():
        if phrase in " ".join(words_in_sentence):
            score += count
    sentence_scores[i] = score
sorted_scores = sorted(sentence_scores.items(), key=lambda item:
item[1], reverse=True)
top_sentence_indices = [index for index, score in sorted_scores[:3]]
top_sentence_indices.sort()
summary_sentences = [sentences[i] for i in top_sentence_indices]
print("Original Paragraph:", para)
print("Summary:", end = " ")
print(" ".join(summary_sentences))
```

```
Name: Sidhanta Barik, RegNo: 2241002049
--------------------------------------------------
Original Text: Hello World
French: Bonjour le monde
Spanish: Hola Mundo
```

```
German: Hallo Welt

Original Paragraph: Greetings World. I am Sid. I am happy to learn
Python. I am sad today.
Sentence 1: Greetings World.
Sentiment: Sentiment(polarity=0.0, subjectivity=0.0)

Sentence 2: I am Sid.
Sentiment: Sentiment(polarity=0.0, subjectivity=0.0)

Sentence 3: I am happy to learn Python.
Sentiment: Sentiment(polarity=0.8, subjectivity=1.0)

Sentence 4: I am sad today.
Sentiment: Sentiment(polarity=-0.5, subjectivity=1.0)

POS Tagging: [('Love', 'VB'), ('never', 'RB'), ('came', 'VBD'), ('to',
'TO'), ('me', 'PRP'), ('in', 'IN'), ('a', 'DT'), ('silver', 'NN'),
('spoon', 'NN'), ('so', 'IN'), ('i', 'JJ'), ('licked', 'VBD'), ('it',
'PRP'), ('off', 'RP'), ('knives', 'NNS')]

The word 'Theyr' is misspelled. Top 3 suggestions are: Her, They,
Their

Paragraph: Greetings World. I am Sid. I am happy to learn Python. I am
sad today.
Adjectives in order of occurrence:
happy

Original Paragraph: Greetings World. I am Sid. I am happy to learn
Python. I am sad today.
Summary: Greetings World. I am Sid. I am happy to learn Python.
```

Q12) Write a Python program that takes a word as input and returns:

- Its definition
- Its synonyms
- Its antonyms(if available)

```
print('''Name: Sidhanta Barik, RegNo: 2241002049
-------------------------------------------------''')
from textblob import Word

wrd = input("Enter a word: ")
w = Word(wrd)
print(f"Word: {wrd}")
defn = w.definitions
print(f"Definitions: {defn}")
synsets = w.synsets
syn = []
```

```
for s in synsets:
    for l in s.lemmas():
        if l.name() not in syn:
            syn.append(l.name())
print(f"Synonyms: {", ".join(syn)}")
ant = []
lemma = synsets[0].lemmas()
antonyms = lemma[0].antonyms()
for a in antonyms:
    ant.append(a.name())
print(f"Antonyms: {", ".join(ant)}")
```

```
Name: Sidhanta Barik, RegNo: 2241002049
--------------------------------------------------
Word: Happy
Definitions: ['enjoying or showing or marked by joy or pleasure',
'marked by good fortune', 'eagerly disposed to act or to be of
service', 'well expressed and to the point']
Synonyms: happy, felicitous, glad, well-chosen
Antonyms: unhappy
```

Q13)

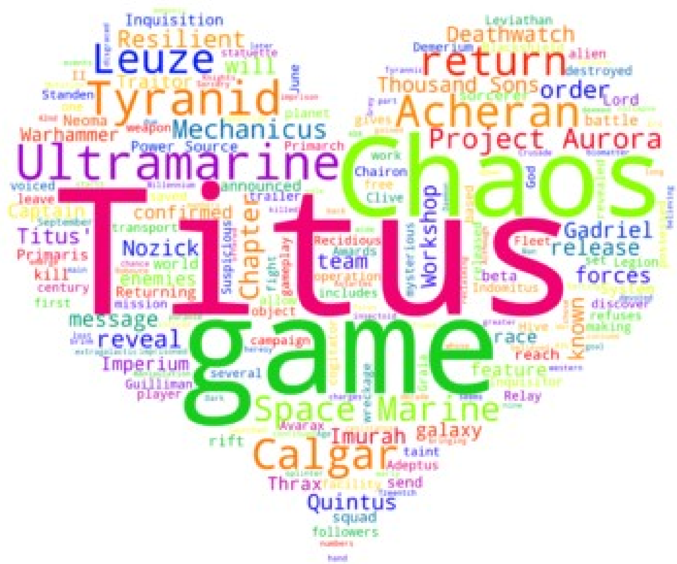
- Write a Python program that reads a .txt file, processes the text, and generates a word cloud visualization.
- Create a word cloud in the shape of an object (e.g., a heart, star) using WordCloud and a mask image.

```
print('''Name: Sidhanta Barik, RegNo: 2241002049
--------------------------------------------------''')
from pathlib import Path
text = Path(r'WarHammer.txt').read_text('UTF-8')

import imageio.v2 as imageio
mask_image = imageio.imread(Path(r'mask_heart.png'))

from wordcloud import WordCloud
import matplotlib.pyplot as plt
wc = WordCloud(colormap = 'prism', mask = mask_image, background_color
= 'white')
wc = wc.generate(text)
plt.imshow(wc, interpolation='bilinear')
plt.axis("off")
plt.show()
```

```
Name: Sidhanta Barik, RegNo: 2241002049
--------------------------------------------------
```



Q14) (Textatistic: Readability of News Articles) Using the above techniques, download from several news sites current news articles on the same topic. Perform readability assessments on them to determine which sites are the most readable. For each article, calculate the average number of words per sentence, the average number of characters per word and the average number of syllables per word.

```
print('''Name: Sidhanta Barik, RegNo: 2241002049
-----------------------------------------------------''')
import requests
from textatistic import Textatistic

url = "https://www.bbc.com/news"
response = requests.get(url)
text = requests.get(url).text[:2000]
blob = TextBlob(text)
readability = Textatistic(text)
print(readability.dict())
```

```
Name: Sidhanta Barik, RegNo: 2241002049
-----------------------------------------------------
{'char_count': 1832, 'word_count': 200, 'sent_count': 22,
'sybl_count': 380, 'notdalechall_count': 122, 'polysyblword_count':
39, 'flesch_score': 36.86772727272731, 'fleschkincaid_score':
10.375454545454549, 'gunningfog_score': 11.436363636363637,
```

```
'smog_score': 10.735267742514802, 'dalechall_score':
13.719309090909091}
```

Q15) (spaCy: Named Entity Recognition) Using the above techniques, download a current news article, then use the spaCy library's named entity recognition capabilities to display the named entities(people, places, organizations, etc.) in the article.

```
print('''Name: Sidhanta Barik, RegNo: 2241002049
-----------------------------------------------------''')

import spacy
nlp = spacy.load('en_core_web_sm')

from pathlib import Path
text = Path(r'currNews.txt').read_text('UTF-8')
doc = nlp(text)
ents = doc.ents
people, places, org = [], [], []
for e in doc.ents:
    if e.label_ == "PERSON":
        people.append(e)
    elif e.label_ == "GPE":
        places.append(e)
    elif e.label_ == "ORG":
        org.append(e)
print("People:", people)
print("Places:", places)
print("Organizations:", org)
```

```
Name: Sidhanta Barik, RegNo: 2241002049
-----------------------------------------------------
People: [Rebecca Henschke, Zhu Minyun, Win Myat Aye]
Places: [Myanmar, Kyaukse, Myanmar, Thailand, Bangkok, China, Myanmar,
Myanmar, Myanmar, Myanmar, Myanmar, Myanmar]
Organizations: [UN, BBC, BBC, Htet Naing Zhaw, BBC, Blue Sky Rescue
Team, the National Unity Government, NUG, BBC]
```

Q16) (spaCy: Shakespeare Similarity Detection) Using the spaCy techniques, download a Shakespeare comedy from Project Gutenberg and compare it for similarity with Romeo and Juliet.

```
print('''Name: Sidhanta Barik, RegNo: 2241002049
-----------------------------------------------------''')
import spacy
import requests
```

```
nlp = spacy.load("en_core_web_sm")
def get_text(url):
    return requests.get(url).text[:5000]
comedy_text =
get_text("https://www.gutenberg.org/cache/epub/22337/pg2233")
rj_text =
get_text("https://www.gutenberg.org/cache/epub/1513/pg1513.txt")
similarity = nlp(comedy_text).similarity(nlp(rj_text))
print(f"Similarity: {similarity:.2f}")
```

```
Name: Sidhanta Barik, RegNo: 2241002049
--------------------------------------------------
Similarity: 0.56

C:\Users\Sidhant Barik\AppData\Local\Temp\
ipykernel_1988\832649335.py:10: UserWarning: [W007] The model you're
using has no word vectors loaded, so the result of the Doc.similarity
method will be based on the tagger, parser and NER, which may not give
useful similarity judgements. This may happen if you're using one of
the small models, e.g. `en_core_web_sm`, which don't ship with word
vectors and only use context-sensitive tensors. You can always add
your own word vectors, or use one of the larger models instead if
available.
  similarity = nlp(comedy_text).similarity(nlp(rj_text))
```

Q17) (textblob.utils Utility Functions) Use strip punc and lowerstrip functions of TextBlob's textblob.utilsmodule with all=True keyword argument to remove punctuation and to get a string in all lowercase letters with whitespace and punctuation removed. Experiment with each function on Romeo and Juliet.

```
print('''Name: Sidhanta Barik, RegNo: 2241002049
--------------------------------------------------''')
from textblob.utils import strip_punc, lowerstrip
import requests
url = "https://www.gutenberg.org/cache/epub/64317/pg64317.txt"
text = requests.get(url).text[3000:3500]
stripped_text = strip_punc(text, all=True)
lower_text = lowerstrip(text, all=True)
print("Stripped:", stripped_text[:100])
print("\nLowercase:", lower_text[:100])
```

```
Name: Sidhanta Barik, RegNo: 2241002049
--------------------------------------------------
Stripped: his way of my tolerance I come to the admission
that it has a limit Conduct may be founded on the h

Lowercase: his way of my tolerance i come to the admission
that it has a limit conduct may be founded on the h
```