Python for Computer Science and Data Science 2 (CSE 3652)

MINOR ASSIGNMENT-3: NATURAL LANGUAGE PROCESSING

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

Ans:- Natural Language Processing (NLP) is a field of artificial intelligence (AI) that enables computers to understand, interpret, and respond to human language in a way that is both meaningful and useful. It combines computational linguistics, machine learning, and deep learning to process and analyze large amounts of natural language data.

Three Real-World Applications of NLP:

a) Chat-bots and Virtual Assistants (e.g., Siri, Alexa, Google Assistant):
Impact: These tools enhance convenience by handling tasks like setting reminders, answering questions, and controlling smart home devices. They make technology accessible to non-technical users and support people with disabilities.

b) Sentiment Analysis (e.g., Monitoring Social Media):
Impact: Businesses use sentiment analysis to understand public opinion, gauge customer satisfaction, and manage brand reputation. It helps in decision-making by providing insights into market trends and consumer behavior.

c) Language Translation (e.g., Google Translate):
Impact: Real-time translation breaks language barriers, facilitating communication in international business, travel, and education. It promotes cultural exchange and helps people understand content in different languages.

NLP transforms how we interact with technology, making machines more intuitive and responsive to human needs.

2. The Explain the following terms and their significance in NLP:
   - Tokenization
   - Stemming
   - Lemmatization

Ans:- a) Tokenization:
Definition: Tokenization is the process of breaking down text into smaller units called tokens. These tokens can be words, phrases, or even characters. For example, the sentence "NLP is fascinating" becomes ["NLP", "is", "fascinating"] after tokenization.
Significance: It's a fundamental preprocessing step in NLP. Tokenization helps in analyzing text by isolating meaningful components, enabling tasks like text analysis, search, and machine translation.

b) Stemming:
Definition: Stemming reduces words to their root or base form by removing suffixes. It uses simple rules without considering the word's meaning. For instance, "running," "runner," and "runs" may all be reduced to "run."
Significance: Stemming speeds up NLP processes by normalizing words, but it may produce non-real words (e.g., "studies" to "studi"). It's useful for applications where accuracy is less critical, like search engines.

d) Lemmatization:
Definition: Lemmatization reduces words to their base or dictionary form (lemma) while considering context and grammar. For example, "running" becomes "run," and "better" becomes "good."
Significance: It's more accurate than stemming, preserving the meaning of words. Lemmatization is preferred in NLP tasks requiring semantic understanding, such as machine translation and sentiment analysis.

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

Ans:- Part-of-Speech (POS) Tagging is a fundamental step in Natural Language Processing (NLP) where each word in a sentence is labeled with its grammatical role, such as noun, verb, adjective, adverb, etc. The process helps machines understand the syntactic structure of sentences.

Importance of POS Tagging:

a) Text Analysis and Understanding: POS tagging enables deeper linguistic analysis, helping machines grasp

context and meaning in text.

b) Named Entity Recognition (NER): Identifying proper nouns (e.g., names, places) becomes easier with accurate POS tagging.

c) Syntactic Parsing: Helps in identifying grammatical relationships and sentence structure.

d) Machine Translation and Chat-bots: Ensures accurate translation and response generation by recognizing the role of each word.

Example: Consider the sentence: "She enjoys a long walk in the morning."

After POS tagging, it would look like:
She (PRON) - Pronoun
enjoys (VERB) - Verb
a (DET) - Determiner
long (ADJ) - Adjective
walk (NOUN) - Noun
in (PREP) - Preposition
the (DET) - Determiner
morning (NOUN) - Noun

4. Create a TextBlob named exercise blob containing "This is a TextBlob".

Ans:-

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

Output:- This is a TextBlob

5. 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."

Ans:-

```
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer, WordNetLemmatizer
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."
words = word_tokenize(text)
sentences = sent_tokenize(text)
ps = PorterStemmer()
lemmatizer = WordNetLemmatizer()
stemmed_words = [ps.stem(word) for word in words]
lemmatized_words = [lemmatizer.lemmatize(word) for word in words]
stop_words = set(stopwords.words('english'))
filtered_words = [word for word in words if word.lower() not in stop_words]
print("Sentences:\n", sentences)
print("Words:\n", words)
print("Stemmed Words:\n", stemmed_words)
print("Lemmatized Words:\n", lemmatized_words)
print("Filtered Words (No Stopwords):\n", filtered_words)
```

Output:-

```
Sentences:
 ['Natural Language Processing enables machines to understand and process human languages.', 'It is a fascinating field with numer
ous 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', 'trans
lation', '.']
Stemmed Words:
 ['natur', 'languag', 'process', 'enabl', 'machin', 'to', 'understand', 'and', 'process', 'human', 'languag', '.', 'it', 'is', 'a'
, 'fascin', 'field', 'with', 'numer', 'applic', ',', 'such', 'as', 'chatbot', 'and', 'languag', 'translat', '.']
Lemmatized Words:
 ['Natural', 'Language', 'Processing', 'enables', 'machine', 'to', 'understand', 'and', 'process', 'human', 'language', '.', 'It',
 'is', 'a', 'fascinating', 'field', 'with', 'numerous', 'application', ',', 'such', 'a', 'chatbots', 'and', 'language', 'translati
```

```
on', '.']
Filtered Words (No Stopwords):
 ['Natural', 'Language', 'Processing', 'enables', 'machines', 'understand', 'process', 'human', 'languages', '.', 'fascinating', '
field', 'numerous', 'applications', ',', 'chatbots', 'language', 'translation', '.']
```

6. 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.

Ans: -

```
import requests
from bs4 import BeautifulSoup
from wordcloud import WordCloud
from nltk.corpus import stopwords
import matplotlib.pyplot as plt
url = "https://www.python.org"
response = requests.get(url)
soup = BeautifulSoup(response.content, "html.parser")
text = soup.get_text(separator=" ")
text = ' '.join(text.split())
stop_words = set(stopwords.words('english'))
filtered_text = ' '.join(word for word in text.split() if word.lower() not in stop_words)
wordcloud = WordCloud(width=800, height=400, background_color='white').generate(filtered_text)
plt.figure(figsize=(10,5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

Output -

7. (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.

Ans:-

```
from wordcloud import WordCloud
from textblob import TextBlob
from bs4 import BeautifulSoup
import requests
from nltk.corpus import stopwords
url = 'https://www.python.org'
response = requests.get(url)
page_content = response.text
soup = BeautifulSoup(page_content, 'html.parser')
text = soup.get_text(separator=' ')
stop_words = set(stopwords.words('english'))
filtered_text = ' '.join(word for word in text.split() if word.lower() not in stop_words)
wordcloud = WordCloud(width=800, height=400, background_color='white').generate(filtered_text)
blob = TextBlob(filtered_text)
```

```
print("\nSentences:")
for sentence in blob.sentences:
    print(sentence)
print("\nWords:")
for word in blob.words:
    print(word)
print("\nNoun Phrases:")
for phrase in blob.noun_phrases:
    print(phrase)
```

Output:-

```
Sentences:
Welcome Python.org Notice: JavaScript essential website, interaction content limited.
Please turn JavaScript full experience.
Skip content ▼ Close Python PSF Docs PyPI Jobs Community ▲ Python Network Donate ≡ Menu Search Site GO Smaller Larger Reset Social
ize LinkedIn Mastodon Chat IRC Twitter Applications Quotes Getting Started Help Python Brochure Downloads releases Source code Win

Words:      Noun Phrases:
Welcome     welcome python.org notice
Python.org  javascript
Notice      essential website
JavaScript  interaction content
essential   please
website     javascript
interaction full experience
content     skip
limited     content ▼
```

8. (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 .

Ans:-

```
from wordcloud import WordCloud
from textblob import TextBlob
from bs4 import BeautifulSoup
import requests
from nltk.corpus import stopwords
url = 'https://www.python.org'
response = requests.get(url)
page_content = response.text
soup = BeautifulSoup(page_content, 'html.parser')
text = soup.get_text(separator=' ')
stop_words = set(stopwords.words('english'))
filtered_text = ' '.join(word for word in text.split() if word.lower() not in stop_words)
wordcloud = WordCloud(width=800, height=400, background_color='white').generate(filtered_text)
blob = TextBlob(filtered_text)
print("\nOverall Sentiment:")
print(blob.sentiment)
print("\nSentiment for Each Sentence:")
for sentence in blob.sentences:
    print(f"{sentence} -> Sentiment: {sentence.sentiment}")
```

Output:-

```
Overall Sentiment:
Sentiment(polarity=0.2339296757664105, subjectivity=0.4451122007244456)

Sentiment for Each Sentence:
Welcome Python.org Notice: JavaScript essential website, interaction content limited. -> Sentiment: Sentiment(polarity=0.242857142
85714288, subjectivity=0.4476190476190476)
Please turn JavaScript full experience. -> Sentiment: Sentiment(polarity=0.35, subjectivity=0.55)
```

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

Ans:-

```
from textblob import TextBlob
from textblob.sentiments import NaiveBayesAnalyzer
from bs4 import BeautifulSoup
import requests
from nltk.corpus import stopwords
url = 'https://www.bbc.com/news/world-asia-60565556'
```

```
response = requests.get(url)
page_content = response.text
soup = BeautifulSoup(page_content, 'html.parser')
text = soup.get_text(separator=' ')
stop_words = set(stopwords.words('english'))
filtered_text = ' '.join(word for word in text.split() if word.lower() not in stop_words)
blob = TextBlob(filtered_text, analyzer=NaiveBayesAnalyzer())
print("\nOverall Sentiment:")
print(blob.sentiment)
print("\nSentiment for Each Sentence:")
for sentence in blob.sentences:
    print(f"{sentence}\nSentiment: {sentence.sentiment}")
```

Output:-

```
Overall Sentiment:
Sentiment(classification='pos', p_pos=1.0, p_neg=8.74874309348703e-23)

Sentiment for Each Sentence:
BBC Skip content British Broadcasting Corporation Home News Sport Business Innovation Culture Arts Travel Earth Audio Video Live H
ome News Israel-Gaza War War Ukraine US & Canada UK UK Politics England N. Ireland N. Ireland Politics Scotland Scotland Politics
Wales Wales Politics Africa Asia China India Australia Europe Latin America Middle East Pictures BBC InDepth BBC Verify Sport Busi
ness Executive Lounge Technology Business Future Business Innovation Technology Science & Health Artificial Intelligence AI v Mind
 Culture Film & TV Music Art & Design Style Books Entertainment News Arts Arts Motion Travel Destinations Africa Antarctica Asia A
ustralia Pacific Caribbean & Bermuda Central America Europe Middle East North America South America World's Table Culture & Experi
ences Adventures SpeciaList Earth Natural Wonders Weather & Science Climate Solutions Sustainable Business Green Living Audio Podc
asts Radio Audio FAQs Video Live Live News Live Sport Home News Sport Business Innovation Culture Arts Travel Earth Audio Video Li
ve Weather Newsletters Error 404 - Oops, page looking longer Try searching instead can't find page looking for.
Sentiment: Sentiment(classification='pos', p_pos=0.9999999957003028, p_neg=4.299686510569712e-09)
typed URL browser, check entered correctly.
Sentiment: Sentiment(classification='pos', p_pos=0.7471442170395055, p_neg=0.2528557829604965)
```

10. Use (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.

Ans:-

```
from textblob import TextBlob
import requests
url = "https://www.gutenberg.org/files/1342/1342-0.txt"
response = requests.get(url)
book_content = response.text
blob = TextBlob(book_content)
words = blob.words
print("\nMisspelled Words and Corrections:")
for word in words:
    if not word.correct() == word:
        print(f"{word} -> {word.correct()}")
```

Output:-

```
Misspelled Words and Corrections:
OF -> of
Reading -> Leading
Jane -> Lane
Jane -> Lane
Austen -> Listen
Ruskin -> Skin
Charing -> Sharing
```

11. • 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 paragraph 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 paragraph by keeping only the most informative sentences, based on noun phrase frequency.

Ans:-

```
from textblob import TextBlob
from collections import Counter
from deep_translator import GoogleTranslator
def translate_text(text):
    print("French:", GoogleTranslator(source='auto', target='fr').translate(text))
    print("Spanish:", GoogleTranslator(source='auto', target='es').translate(text))
    print("German:", GoogleTranslator(source='auto', target='de').translate(text))
def analyze_sentiment(text):
    blob = TextBlob(text)
    for sentence in blob.sentences:
        print(f"{sentence} -> Polarity: {sentence.sentiment.polarity}, Subjectivity: {sentence.sentiment.subjectivity}")
def sentence_sentiment(paragraph):
    blob = TextBlob(paragraph)
    for sentence in blob.sentences:
        print(f"{sentence} -> Sentiment Score: {sentence.sentiment}")
def pos_tagging(sentence):
    blob = TextBlob(sentence)
    for word, tag in blob.tags:
        print(f"{word} -> {tag}")
def spell_check(word):
    blob = TextBlob(word)
    suggestions = blob.correct()
    print("Suggested Correction:", suggestions)
def extract_adjectives(paragraph):
    blob = TextBlob(paragraph)
    adjectives = [word for word, tag in blob.tags if tag == "JJ"]
    print("Adjectives:", adjectives)
def extract_keywords(text):
    blob = TextBlob(text)
    noun_phrases = blob.noun_phrases
    common_phrases = Counter(noun_phrases).most_common(5)
    print("Top 5 Keywords:", common_phrases)
def summarize_text(paragraph):
    blob = TextBlob(paragraph)
    scores = {sentence: sum(blob.noun_phrases.count(phrase) for phrase in sentence.split()) for sentence in blob.sentences}
    summary = sorted(scores, key=scores.get, reverse=True)[:3]
    print("Summary:", ' '.join(str(s) for s in summary))
text = input("Enter a paragraph: ")
translate_text(text)
analyze_sentiment(text)
sentence_sentiment(text)
pos_tagging(text)
word = input("Enter a word to check spelling: ")
spell_check(word)
extract_adjectives(text)
extract_keywords(text)
summarize_text(text)
```

Output: -

```
Enter a paragraph: Hello how are you?
French: Bonjour comment allez-vous?
Spanish: ¿Hola, cómo estás?
German: Hallo, wie geht es dir?
Hello how are you? -> Polarity: 0.0, Subjectivity: 0.0
Hello how are you? -> Sentiment Score: Sentiment(polarity=0.0, subjectivity=0.0)
Hello -> NNP
how -> WRB
are -> VBP
you -> PRP
Enter a word to check spelling: bonjour
Suggested Correction: bonjour
Adjectives: []
Top 5 Keywords: [('hello', 1)]
Summary: Hello how are you?
```

12. Write a Python program that takes a word as input and returns:
    - Its definition
    - Its synonyms
    - Its antonyms(if available)

Ans: -

```
from nltk.corpus import wordnet
def word_analysis(word):
    synonyms = []
    antonyms = []
    definitions = []
    for syn in wordnet.synsets(word):
        definitions.append(syn.definition())
        for lemma in syn.lemmas():
            synonyms.append(lemma.name())
            if lemma.antonyms():
                antonyms.append(lemma.antonyms()[0].name())
    print("Definitions:", definitions)
    print("Synonyms:", set(synonyms))
    print("Antonyms:", set(antonyms))
word = input("Enter a word: ")
word_analysis(word)
```

Output -

```
Enter a word: hello
Definitions: ['an expression of greeting']
Synonyms: {'hello', 'hullo', 'hi', 'how-do-you-do', 'howdy'}
Antonyms: set()
```

13. • 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.

Ans: -

```
from wordcloud import WordCloud
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
def generate_wordcloud(file_path):
    with open(file_path, 'r') as file:
        text = file.read()
    wordcloud = WordCloud(width=800, height=400, background_color='white').generate(text)
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis('off')
    plt.show()
def generate_shaped_wordcloud(file_path, mask_path):
    mask = np.array(Image.open(mask_path))
    with open(file_path, 'r') as file:
        text = file.read()
    wordcloud = WordCloud(mask=mask, background_color='white', contour_width=1, contour_color='black').generate(text)
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis('off')
    plt.show()
generate_wordcloud('file.txt')
generate_shaped_wordcloud('file.txt', 'image.png')
```

Output -

14. (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.

Ans: -

```
import requests
from bs4 import BeautifulSoup
import textstat
urls = [
    "https://www.bbc.com/news/world",
    "https://edition.cnn.com/world",
    "https://www.aljazeera.com/news/"
]
def analyze_readability(url):
    response = requests.get(url)
    soup = BeautifulSoup(response.content, 'html.parser')
    text = ' '.join([p.get_text() for p in soup.find_all('p')])
    readability = {
        'avg_words_per_sentence': textstat.avg_sentence_length(text),
        'avg_chars_per_word': textstat.avg_character_per_word(text),
        'avg_syllables_per_word': textstat.avg_syllables_per_word(text)
    }
    print("URL:", url)
    print("Average Words per Sentence:", readability['avg_words_per_sentence'])
    print("Average Characters per Word:", readability['avg_chars_per_word'])
    print("Average Syllables per Word:", readability['avg_syllables_per_word'])
for url in urls:
    analyze_readability(url)
```

Output -

```
URL: https://www.bbc.com/news/world
Average Words per Sentence: 15.3
Average Characters per Word: 5.16
Average Syllables per Word: 1.6
URL: https://edition.cnn.com/world
Average Words per Sentence: 10.0
Average Characters per Word: 4.43
Average Syllables per Word: 1.3
URL: https://www.aljazeera.com/news/
Average Words per Sentence: 16.9
Average Characters per Word: 5.22
Average Syllables per Word: 1.7
```

15. (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.

Ans: -

```
import requests
from bs4 import BeautifulSoup
import spacy
nlp = spacy.load('en_core_web_sm')
def analyze_entities(url):
    response = requests.get(url)
    soup = BeautifulSoup(response.content, 'html.parser')
    text = ' '.join([p.get_text() for p in soup.find_all('p')])
    doc = nlp(text)
    for ent in doc.ents:
        print(f"{ent.text} -> {ent.label_}")
url = input("Enter the news article URL: ")
analyze_entities(url)
```

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

Ans: -

```
import spacy
from urllib.request import urlopen
nlp = spacy.load('en_core_web_md')
def fetch_text(url):
    response = urlopen(url)
    return response.read().decode('utf-8')
romeo_juliet_url = "https://www.gutenberg.org/cache/epub/1513/pg1513.txt"
comedy_url = "https://www.gutenberg.org/cache/epub/1524/pg1524.txt"
romeo_juliet_text = nlp(fetch_text(romeo_juliet_url))
```

```
comedy_text = nlp(fetch_text(comedy_url))
similarity_score = romeo_juliet_text.similarity(comedy_text)
print(f"Similarity between Romeo and Juliet and the comedy: {similarity_score:.2f}")
```

17. (textblob.utils Utility Functions) Use strip punc and lowerstrip functions of TextBlob'stextblob.utils module 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.

Ans: -

```
from textblob.utils import strip_punc, lowerstrip
from urllib.request import urlopen
url = "https://www.gutenberg.org/cache/epub/1513/pg1513.txt"
response = urlopen(url)
text = response.read().decode('utf-8')
cleaned_text_strip = strip_punc(text, all=True)
cleaned_text_lowerstrip = lowerstrip(text, all=True)
print("After strip_punc:\n", cleaned_text_strip[:500])
print("\nAfter lowerstrip:\n", cleaned_text_lowerstrip[:500])
```

Output: -

```
After strip_punc:
 The Project Gutenberg eBook of Romeo and Juliet

This ebook is for the use of anyone anywhere in the United States and
most other parts of the world at no cost and with almost no restrictions
whatsoever You may copy it give it away or reuse it under the terms
of the Project Gutenberg License included with this ebook or online
at wwwgutenbergorg If you are not located in the United States
you will have to check the laws of the country where you are located
before using this eBook



After lowerstrip:
 the project gutenberg ebook of romeo and juliet

this ebook is for the use of anyone anywhere in the united states and
most other parts of the world at no cost and with almost no restrictions
whatsoever you may copy it give it away or reuse it under the terms
of the project gutenberg license included with this ebook or online
at wwwgutenbergorg if you are not located in the united states
you will have to check the laws of the country where you are located
before using this ebook
```