Name: Neha Vaghela Enrollment No:2101031000256

NLP Practical1 to 4

Practical – 1 Explore different NLP python libraries.

1. NLTK (Natural Language Toolkit)

- **Purpose:** Comprehensive library for NLP tasks, suitable for beginners and research.
- **Example:**

```
import nltk

from nltk.tokenize import word_tokenize

from nltk.sentiment.vader import SentimentIntensityAnalyzer

nltk.download('vader_lexicon')

text = "This is a really great product. I love it!"

words = word_tokenize(text)

print(words)

# Sentiment analysis

analyzer = SentimentIntensityAnalyzer()

sentiment = analyzer.polarity_scores(text)

print(sentiment)
```

- **Output:**

```
['This', 'is', 'a', 'really', 'great', 'product', '.', 'I', 'love', 'it', '!']

{'neg': 0.0, 'neu': 0.318, 'pos': 0.682, 'compound': 0.8316}
```

2. SpaCy

- **Purpose:** Industrial-strength NLP library, known for speed and efficiency.
- **Example:**

```
import spacy
nlp = spacy.load("en_core_web_sm")
doc = nlp("Apple is looking at buying U.K. startup for $1 billion")
for token in doc:
   print(token.text, token.pos_)
# Named Entity Recognition (NER)
for entity in doc.ents:
   print(entity.text, entity.label_)
```

- **Output:** (Depends on the specific model loaded)

3. Gensim

- **Purpose:** Topic modeling, document similarity, and large text corpora handling.
- **Example:**

```
from gensim.corpora import Dictionary
from gensim.models import LdaModel
documents = ["Human machine interface for lab abc computer applications",
             "A survey of user opinion of computer system response time",
             "The EPS user interface management system"]
# Create a dictionary
dictionary = Dictionary(documents)
doc_term_matrix = [dictionary.doc2bow(doc) for doc in documents]
# Train LDA model
lda_model = LdaModel(doc_term_matrix, num_topics=2, id2word=dictionary, passes=15)
```

```
for topic in lda_model.print_topics():
    print(topic)
```

- **Output:** (Depends on the data and model parameters)

4. TextBlob

- **Purpose:** Simple API for common NLP tasks like sentiment analysis, part-of-speech tagging, and more.
- **Example:**

```
from textblob import TextBlob

text = "This movie is amazing!"

blob = TextBlob(text)

print(blob.sentiment)
```

- **Output:** Sentiment(polarity=1.0, subjectivity=1.0)

5. Transformers (Hugging Face)

- **Purpose:** State-of-the-art NLP models for various tasks like text generation, translation, question answering, etc.
- **Example:**

```
from transformers import pipeline

# Sentiment analysis pipeline

nlp = pipeline("sentiment-analysis")

result = nlp("This is a fantastic product!")

print(result)
```

- **Output:** [{'label': 'POSITIVE', 'score': 0.9999999403953552}]

Practical – 2 Implement tokenization, stemming, and lemmatization on a sample text.

Input:

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer

# Sample text
text = "Natural language processing (NLP) is a field of artificial intelligence that gives computers the ability to understand text and spoken words in much the same way human beings can."

# Tokenization
tokens = word_tokenize(text)
print("Tokens:", tokens)

# Stemming
stemmer = PorterStemmer()
stems = [stemmer.stem(token) for token in tokens]
print("Stems:", stems)

# Lemmatization
lemmatizer = WordNetLemmatizer()
# Note: Lemmatization requires part of speech tagging for accurate results
lemmatized = [lemmatizer.lemmatize(token) for token in tokens]
```

```
print("Lemmatized:", lemmatized)
```

Output:

```
Tokens:

['Natural', 'language', 'processing', '(', 'NLP', ')', 'is', 'a', 'field', 'of', 'artificial',
'intelligence', 'that', 'gives', 'computers', 'the', 'ability', 'to', 'understand',
'text', 'and', 'spoken', 'words', 'in', 'much', 'the', 'same', 'way', 'human',
'beings', 'can', '.']

Stems:

['natur', 'languag', 'process', '(', 'nlp', ')', 'is', 'a', 'field', 'of', 'artifici', 'intellig',
'that', 'give', 'comput', 'the', 'abil', 'to', 'understand', 'text', 'and', 'spoken',
'word', 'in', 'much', 'the', 'same', 'way', 'human', 'be', 'can', '.']

Lemmatized:

['Natural', 'language', 'processing', '(', 'NLP', ')', 'is', 'a', 'field', 'of', 'artificial',
'intelligence', 'that', 'give', 'computer', 'the', 'ability', 'to', 'understand', 'text',
'and', 'spoken', 'word', 'in', 'much', 'the', 'same', 'way', 'human', 'being',
'can', '.']
```

Practical - 3 Write regular expressions to extract dates, email addresses, and phone numbers from

a given text.

Input:

Sample Text

"Please contact John at john.doe@example.com on 12/31/2023 or call him at 123-456-7890. Alternatively, you can reach him at jane.doe@example.org or on 01-01-2024 at 987.654.3210."

Regular Expressions

1. Extract Dates

```
import re

text = "Please contact John at john.doe@example.com on 12/31/2023 or call him at 123-456-7890. Alternatively, you can reach him at jane.doe@example.org or on 01-01-2024 at 987.654.3210."

date_pattern = r'\b(?:\d{1,2}[-/]\d{1,2}[-/]\d{2,4}|\d{4}[-/]\d{1,2}[-/]\d{1,2})\b'

dates = re.findall(date_pattern, text)

print("Dates:", dates)
```

2. Extract Email Addresses

```
email_pattern = r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b'

emails = re.findall(email_pattern, text)

print("Emails:", emails)
```

3. Extract Phone Numbers

```
phone_pattern = r'\b\d{3}[-.\s]?\d{3}[-.\s]?\d{4}\b'

phones = re.findall(phone_pattern, text)

print("Phone Numbers:", phones)
```

Output:

```
Dates:

['12/31/2023', '01-01-2024']

Emails:

['john.doe@example.com', 'jane.doe@example.org']

Phone Numbers:

['123-456-7890', '987.654.3210']
```

Practical - 4 Practice pattern matching and text cleaning using Python's re library.

Input:

1. Extract Email Addresses

```
import re

text = "Contact us at info@example.com or visit https://www.example.com for more info. Call 123-456-7890."

email_pattern = r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b'

emails = re.findall(email_pattern, text)

print("Emails:", emails)
```

2. Extract URLs

```
url_pattern = r'http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\\(\\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+'

urls = re.findall(url_pattern, text)

print("URLs:", urls)
```

3. Extract Phone Numbers

```
phone_pattern = r'\b\d{3}[-.]\d{3}[-.]\d{4}\b'

phones = re.findall(phone_pattern, text)

print("Phone Numbers:", phones)
```

4. Remove Email Addresses, URLs, and Phone Numbers

```
cleaned_text = re.sub(email_pattern, '', text)

cleaned_text = re.sub(url_pattern, '', cleaned_text)

cleaned_text = re.sub(phone_pattern, '', cleaned_text)

print("Cleaned Text:", cleaned_text)
```

Output:

```
Emails :

['info@example.com']

URLs:

['https://www.example.com']
```

```
Phone Numbers:
['123-456-7890']
Cleaned Text:
"Contact us at  or visit  for more info. Call ."
```