Experiment - 7: Perform the steps involved in Text Analytics in Python & R

Task to be performed :

Explore Top-5 Text Analytics Libraries in Python (w.r.t Features & Applications)

Explore Top-5 Text Analytics Libraries in R (w.r.t Features & Applications)

Perform the following experiments using Python & R

Tokenization (Sentence & Word)

Frequency Distribution

Remove stopwords & punctuations

Lexicon Normalization (Stemming, Lemmatization)

Part of Speech tagging

Named Entity Recognization

Scrape data from a website

Explore Top-5 Text Analytics Libraries in Python

1. **NLTK (Natural Language Toolkit):**
    - **Features:**
        - Comprehensive set of libraries and tools for natural language processing (NLP).
        - Tokenization, stemming, tagging, parsing, and semantic reasoning functionalities.
        - Supports various corpora and lexical resources.
        - Provides interfaces to popular resources like WordNet.
    - **Applications:**
        - Text classification and sentiment analysis.
        - Named entity recognition.
        - Part-of-speech tagging.
        - Concordance and collocation analysis.
2. **Scattertext:**
    - **Features:**
        - Specifically designed for visualizing linguistic variation between document categories.

        - Produces interactive scatter plots that highlight terms differentiating categories.
        - Supports customization and interactive exploration of visualizations.
        - Handles linguistic and stylistic differences well.
    - **Applications:**
        - Comparative analysis of document categories.
        - Identifying distinctive terms in different contexts.
        - Visual exploration of language patterns.
3. **SpaCy:**
    - **Features:**
        - Fast and efficient NLP library.
        - Tokenization, part-of-speech tagging, named entity recognition, and dependency parsing.
        - Pre-trained models for various languages.
        - Easy integration with machine learning pipelines.
    - **Applications:**
        - Named entity recognition and extraction.
        - Dependency parsing for understanding relationships between words.
        - Text summarization.
        - Information extraction.
4. **TextBlob:**
    - **Features:**
        - Simple and intuitive API for common NLP tasks.
        - Built on top of NLTK and Pattern libraries.
        - Part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more.
        - Easy to use for beginners.
    - **Applications:**
        - Sentiment analysis and classification.
        - Basic text processing tasks.
        - Language translation.
        - Parsing and extracting information from text.
5. **scikit-learn (sklearn):**
    - **Features:**

    - General-purpose machine learning library with text processing capabilities.
    - Text vectorization techniques (TF-IDF, CountVectorizer).
    - Integration with other machine learning algorithms for text classification and clustering.
    - Comprehensive documentation and community support.
  - **Applications:**
    - Text classification (e.g., spam detection).
    - Clustering and topic modeling.
    - Feature extraction and representation.
    - Text regression.

Explore Top-5 Text Analytics Libraries in R

1. **shiny:**
   - **Features:**
     - Web application framework for R.
     - Allows for the creation of interactive and dynamic web-based dashboards and applications.
     - Well-suited for building user interfaces and visualizations for text analytics applications.
     - Integration with other R libraries for data processing and analysis.
   - **Applications:**
     - Building interactive dashboards for text analysis results.
     - Creating user-friendly interfaces for exploring and visualizing text data.
     - Incorporating text analytics into web-based applications.
2. **tm (Text Mining Package):**
   - **Features:**
     - Comprehensive package for text mining in R.
     - Supports text preprocessing tasks such as cleaning, stemming, and stopword removal.
     - Provides functions for creating document-term matrices (DTM) and term-document matrices (TDM).
     - Integration with other R packages for statistical analysis.
   - **Applications:**

        - Document clustering and classification.
        - Term frequency analysis.
        - Text preprocessing and transformation.
        - Integration with machine learning algorithms for text analysis.

3. **quanteda:**
    - **Features:**
        - Modern and flexible package for quantitative text analysis.
        - Supports corpus management, document-feature matrices, and various text analysis operations.
        - Designed for efficiency and scalability in handling large text datasets.
        - Integration with other R packages for statistical analysis and visualization.
    - **Applications:**
        - Document-feature matrix creation for text analysis.
        - Text preprocessing, including tokenization and stemming.
        - Sentiment analysis and text classification.
        - Topic modeling and exploratory data analysis.
4. **quanteda.textstats:**
    - **Features:**
        - An extension of the quanteda package, specifically focusing on text statistics.
        - Provides functions for calculating various text statistics, such as word frequencies, lexical diversity, and readability measures.
        - Useful for gaining insights into the linguistic characteristics of a text corpus.
        - Complements quanteda's core functionalities for text analysis.
    - **Applications:**
        - Analyzing word frequencies and patterns in a corpus.
        - Assessing the complexity and readability of text.
        - Extracting key statistical information about a text dataset.
5. **tm.plugin.sentiment:**
    - **Features:**
        - A plugin for the tm package that focuses on sentiment analysis.
        - Enables sentiment analysis on text data by incorporating pre-trained sentiment lexicons.
        - Supports the calculation of sentiment scores for individual documents or terms.

        - Useful for understanding the emotional tone or sentiment expressed in a text corpus.
    - **Applications:**
        - Sentiment analysis in text mining projects.
        - Assessing the sentiment polarity (positive, negative, neutral) of documents.
        - Incorporating sentiment analysis into larger text analytics workflows.

```
pip install nltk beautifulsoup4
```

```
    Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (
    Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.10/dist-
    Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages
    Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages
    Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist
    Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (
    Requirement already satisfied: soupsieve>1.2 in /usr/local/lib/python3.10/dist-p
```

1: Tokenization (Sentence & Word)

```
import nltk
nltk.download('punkt')
```

```
    [nltk_data] Downloading package punkt to /root/nltk_data...
    [nltk_data]   Unzipping tokenizers/punkt.zip.
    True
```

```
import nltk
from nltk.tokenize import word_tokenize, sent_tokenize

# Sample text
text = "NLTK is a powerful library for natural language processing"

# Sentence Tokenization
sentences = sent_tokenize(text)
print("Sentence Tokenization:")
print(sentences)

# Word Tokenization
words = word_tokenize(text)
print("\nWord Tokenization:")
print(words)
```

```
    Sentence Tokenization:
    ['NLTK is a powerful library for natural language processing']
```

```
Word Tokenization:
['NLTK', 'is', 'a', 'powerful', 'library', 'for', 'natural', 'language', 'proces
```

2: Frequency Distribution

```
from nltk import FreqDist

# Sample words
word_list = ["apple", "banana", "apple", "orange", "banana", "apple", "grape"]

# Calculate frequency distribution
freq_dist = FreqDist(word_list)
print("Frequency Distribution:")
print(freq_dist)
```

```
Frequency Distribution:
<FreqDist with 4 samples and 7 outcomes>
```

3: Remove Stopwords & Punctuations

```
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
True
```

```
from nltk.corpus import stopwords
from string import punctuation

# Sample text
text = "This is a sample sentence, with some stopwords and punctuations."

# Remove stopwords and punctuations
stop_words = set(stopwords.words("english"))
filtered_text = [word.lower() for word in word_tokenize(text) if word.isalnum() and

print("Text after removing stopwords and punctuations:")
print(filtered_text)
```

```
Text after removing stopwords and punctuations:
['sample', 'sentence', 'stopwords', 'punctuations']
```

4: Lexicon Normalization (Stemming, Lemmatization)

```
import nltk
nltk.download('wordnet')
```

```
    [nltk_data] Downloading package wordnet to /root/nltk_data...
    True
```

```
from nltk.stem import PorterStemmer, WordNetLemmatizer

# Sample words
words = ["running", "better", "cats", "dogs"]

# Stemming
porter_stemmer = PorterStemmer()
stemmed_words = [porter_stemmer.stem(word) for word in words]
print("Stemmed Words:")
print(stemmed_words)

# Lemmatization
lemmatizer = WordNetLemmatizer()
lemmatized_words = [lemmatizer.lemmatize(word) for word in words]
print("\nLemmatized Words:")
print(lemmatized_words)
```

```
    Stemmed Words:
    ['run', 'better', 'cat', 'dog']

    Lemmatized Words:
    ['running', 'better', 'cat', 'dog']
```

5: Part of Speech Tagging

```
import nltk
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
```

```
# Sample text
text = "NLTK is a powerful library for natural language processing."

# Part of Speech Tagging
pos_tags = nltk.pos_tag(word_tokenize(text))
print("Part of Speech Tagging:")
print(pos_tags)
```

```
Part of Speech Tagging:
[('NLTK', 'NNP'), ('is', 'VBZ'), ('a', 'DT'), ('powerful', 'JJ'), ('library', 'N
```

6: Named Entity Recognition

```
import nltk
nltk.download('maxent_ne_chunker')
nltk.download('words')
```

```
[nltk_data] Downloading package maxent_ne_chunker to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping chunkers/maxent_ne_chunker.zip.
[nltk_data] Downloading package words to /root/nltk_data...
[nltk_data]   Unzipping corpora/words.zip.
True
```

```
nltk.download('words')# Sample text
text = "Barack Obama was the 44th President of the United States."

# Named Entity Recognition
from nltk import ne_chunk
named_entities = ne_chunk(pos_tags)
print("Named Entity Recognition:")
print(named_entities)
```

```
Named Entity Recognition:
(S
  (ORGANIZATION NLTK/NNP)
  is/VBZ
  a/DT
  powerful/JJ
  library/NN
  for/IN
  natural/JJ
  language/NN
  processing/NN
  ./.)
[nltk_data] Downloading package words to /root/nltk_data...
[nltk_data]   Package words is already up-to-date!
```

7: Scrape data from a website

```python
import requests
from bs4 import BeautifulSoup

# URL to scrape
url = "https://example.com"

# Make a request to the URL
response = requests.get(url)

# Parse HTML content
soup = BeautifulSoup(response.text, "html.parser")

# Extract text content from the webpage
webpage_text = soup.get_text()

print("Text extracted from the website:")
print(webpage_text)
```

```
Text extracted from the website:



Example Domain







Example Domain
This domain is for use in illustrative examples in documents. You may use this
    domain in literature without prior coordination or asking for permission.
More information...
```

```
import requests
from bs4 import BeautifulSoup

# URL to scrape (Python official documentation homepage)
url = "https://docs.python.org/3/"

# Make a request to the URL
response = requests.get(url)

# Parse HTML content
soup = BeautifulSoup(response.text, "html.parser")

# Extract text content from the webpage
webpage_text = soup.get_text()

# Display a portion of the extracted text (for brevity)
print("Text extracted from the Python documentation homepage:")
print(webpage_text[:500])
```

```
Text extracted from the Python documentation homepage:




3.12.2 Documentation
```

Theme

Auto
Light
Dark

R

```
install.packages(c("shiny", "tm", "SnowballC", "NLP"))
```

```
Installing packages into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)
```

```
# Install necessary packages
install.packages(c("tokenizers", "tm", "stringr", "SnowballC", "openNLP", "NLP", "rv

# Load required libraries
library(tokenizers)
library(tm)
library(stringr)
library(SnowballC)
library(openNLP)
library(NLP)
library(udpipe)

# Sample text
text <- "This is a sample sentence. Tokenization in R is interesting!"

# Sentence Tokenization
sent_tokens <- sent_token_annotate(text)$features

# Word Tokenization
word_tokens <- word_token_annotate(text)$features

# Frequency Distribution
word_freq <- table(word_tokens)
print(word_freq)

# Remove stopwords and punctuations
stopwords <- stopwords("en")
filtered_tokens <- word_tokens[!(word_tokens %in% stopwords) & !word_tokens %in% str

# Lexicon Normalization (Stemming, Lemmatization)
stemmed_tokens <- wordStem(filtered_tokens)
lemmatized_tokens <- lemmatize_words(filtered_tokens)

# Part of Speech tagging
pos_tags <- pos_tag_annotate(text)$features

# Named Entity Recognition (NER)
# Note: NER requires a pre-trained model, for example, the spaCy model
# You can use udpipe for POS tagging, but for NER, you might want to use spaCy in Py
# Alternatively, you can explore the 'cleanNLP' package for NER in R

# Scraping data from a website
library(rvest)

# Example: Scraping titles from a website
url <- "https://example.com"
webpage <- read_html(url)
titles <- html_text(html_nodes(webpage, "h2"))
print(titles)
```

```
Installing packages into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

also installing the dependencies ‘openNLPdata’, ‘rJava’


Warning message in install.packages(c("tokenizers", "tm", "stringr", "SnowballC"
“installation of package ‘rJava’ had non-zero exit status”
Warning message in install.packages(c("tokenizers", "tm", "stringr", "SnowballC"
“installation of package ‘openNLPdata’ had non-zero exit status”
Warning message in install.packages(c("tokenizers", "tm", "stringr", "SnowballC"
“installation of package ‘openNLP’ had non-zero exit status”
Loading required package: NLP

Error in library(openNLP): there is no package called ‘openNLP’
Traceback:

1. library(openNLP)
```

1. Tokenization (Sentence & Word)

```
# Tokenization (Sentence & Word)
text <- "This is a sample sentence. Tokenization is important for NLP."
sentences <- strsplit(text, "\\.")[[1]]
words <- unlist(strsplit(text, "\\s+"))

print("Sentences:")
print(sentences)
print("Words:")
print(words)
```

```
[1] "Sentences:"
[1] "This is a sample sentence"          " Tokenization is important for NLP"
[1] "Words:"
 [1] "This"         "is"           "a"            "sample"       "sentence."
 [6] "Tokenization" "is"           "important"    "for"          "NLP."
```

```
install.packages("tokenizers")
library(tokenizers)

text <- "This is a sample sentence. Tokenization is important for NLP."
sentences <- tokenize_sentences(text)
words <- tokenize_words(text)

print("Sentences:")
print(sentences)
print("Words:")
print(words)
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

also installing the dependencies ‘Rcpp’, ‘SnowballC’


[1] "Sentences:"
[[1]]
[1] "This is a sample sentence."         "Tokenization is important for NLP."

[1] "Words:"
[[1]]
 [1] "this"         "is"           "a"            "sample"       "sentence"
 [6] "tokenization" "is"           "important"    "for"          "nlp"
```

2. Frequency Distribution

```
# Frequency Distribution
word_freq <- table(words)
print("Word frequency:")
print(word_freq)
```

```
[1] "Word frequency:"
words
           a          for    important           is          nlp       sample
           1            1            1            2            1            1
    sentence         this tokenization
           1            1            1
```

3. Remove stopwords & punctuations

```
# Remove stopwords & punctuations
stop_words <- c("is", "a", "for")  # Example list of stopwords
filtered_words <- words[!tolower(words) %in% stop_words & !grepl("[[:punct:]]", word
print("Filtered words:")
 print(filtered_words)
```

```
[1] "Filtered words:"
list()
```

4. Lexicon Normalization (Stemming, Lemmatization)

```
install.packages("SnowballC")
library(SnowballC)

# Example data
filtered_words <- c("running", "flies", "happily", "jumps")

# Stemming using SnowballC
stemmed_words <- wordStem(filtered_words)

# Print results
print("Stemmed words:")
print(stemmed_words)
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

[1] "Stemmed words:"
[1] "run"     "fli"     "happili" "jump"
```

5. Part of Speech tagging

```
install.packages("udpipe", dependencies=TRUE)
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

also installing the dependencies ‘modeltools’, ‘topicmodels’


Warning message in install.packages("udpipe", dependencies = TRUE):
“installation of package ‘topicmodels’ had non-zero exit status”
```

```
# Install and load the udpipe package
install.packages("udpipe")
library(udpipe)

# Download and load the English model
ud_model <- udpipe_download_model(language = "english")
ud_model <- udpipe_load_model(ud_model$file_model)

# Example data
words <- c("running", "flies", "happily", "jumps")

# Annotate for lemmatization
x <- udpipe_annotate(ud_model, x = words, doc_id = 1:length(words))
lemmatized_words <- as.data.frame(x)$lemma

# Print the result
print("Lemmatized words:")
print(lemmatized_words)
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

Downloading udpipe model from https://raw.githubusercontent.com/jwijffels/udpipe

 - This model has been trained on version 2.5 of data from https://universaldepe

 - The model is distributed under the CC-BY-SA-NC license: https://creativecommo

 - Visit https://github.com/jwijffels/udpipe.models.ud.2.5 for model license det

 - For a list of all models and their licenses (most models you can download wit

Downloading finished, model stored at '/content/english-ewt-ud-2.5-191206.udpipe

[1] "Lemmatized words:"
[1] "run"     "flie"    "happily" "jump"
```

6. Named Entity Recognization

```
install.packages("NLP")
install.packages("openNLP")
library(openNLP)
library(NLP)

ner_tags <- maxent_tagger_chunker(filtered_words, pos_tags)
print("Named Entities:")
print( ner_tags)
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

also installing the dependencies ‘openNLPdata’, ‘rJava’
```