

PRACTICAL-01

Aim:-Scrape an online E-Commerce Site for Data.

1-: Extract product data from Amazon - be it any product and put these details in the MySQL database. One can use pipeline. Like 1 pipeline to process the scraped data and other to put data in the database and since Amazon has some restrictions on scraping of data, ask them to work on small set of requests otherwise proxies and all would have to be used.

2-: Scrape the details like colour, dimensions, material etc. or customer ratings by features.

Method

A powerful form of data collection, web scraping allows you to gather a wide array of information from websites and transfer them to spreadsheets and databases. Although it sounds difficult to the uninitiated, web scraping is actually quite simple if you use an Application Programming Interface (API). Using an API will automate a lot of the repetitive work associated with manual scraping — saving you a lot of time and energy that you can use for other tasks.

Material

- BeautifulSoup
- Requests
- mysqlConnector

Program

```
from bs4 import BeautifulSoup
import requests
import mysql.connector
```

```

mydb =
mysql.connector.connect(host="localhost",user="root",password="system",database="product",a
uth_plugin = 'mysql_native_password')
mycursor = mydb.cursor()
x='y'
while (x=='y' or x=="Y"):
    try:

```

```

        URL = "https://www.amazon.in/Fire-Boltt-Bluetooth-Calling-Assistance-
Resolution/dp/B0BF57RN3K/ref=sr_1_1?pd_rd_r=132b3f0f-626b-4b98-afbc-
c3b6f193c3a5&pd_rd_w=TxdNk&pd_rd_wg=oZeTh&pf_rd_p=22a704c7-cdfe-40ab-ab4e-
31cbf3ad0e01&pf_rd_r=DQ80CW83D44WM55T1DMR&qid=1678875410&sr=8-1"
        HEADERS = ({
            'User-Agent': 'Mozilla/5.0 (X11; Window x86_64) AppleWebKit / 537.36(KHTML, like
Gecko) Chrome / 44.0.2403.157 Safari / 537.36',
            'Accept-Language': 'en-US, en;q=0.5'})
        webpage = requests.get(URL, headers=HEADERS)
        soup1 = BeautifulSoup(webpage.content, "html.parser")
        soup2 = BeautifulSoup(soup1.prettify(),"html.parser")
        title = soup2.find(id='productTitle').get_text()
        price = soup2.find('span',class_='a-price-whole').get_text().lstrip()
        price=price.replace("\n", "")
        price=price.replace(" ", "")
        price=price.replace(".", "")
        price=price.replace(",", "")

        rating = soup2.find('span', attrs={'class':'a-icon-alt'}).get_text().strip()
        rating=rating.replace("\n", "")

```

```
mycursor.execute("INSERT INTO products (name, price) VALUES (%s, %s)", (title[0:35],
int(price)))
```

```
print("Successfull saved the following data in mysql")
```

```
print("Name:",title[0:35])
```

```
print("price:", price)
```

```
print("rating:",rating)
```

```
except:
```

```
    print(Exception)
```

```
finally:
```

```
    x=input("do you wish to put more data y/n")
```

```
mydb.commit()
```

```
mydb.close()
```

```
print(title)
```

```
print(price)
```

OutPut

```

-----
Successfull saved the following data in mysql
name:
    Fire-Boltt Ninja Call
price: 1799
ratings: 4.3 out of 5 stars
do you wish to put more data y/n n

    Fire-Boltt Ninja Call Pro Plus 1.83" Smart Watch with Bluetooth Calling, AI Voice Assistance, 100 Sports Modes IP67 Rating, 240 * 280 Pixel High Resolution
1799

```

```

"help", "copyright", "credits" or "license()" for more information.
MySQL 8.0 Command Line Client - Unicode
mysql> use product;
Database changed
mysql> create table productss;
ERROR 4028 (HY000): A table must have at least one visible column.
mysql> create table productss(name varchar(1000),price varchar(1000),ratings varchar(1000));
Query OK, 0 rows affected (0.08 sec)

mysql> select * from productss;
Empty set (0.00 sec)

mysql> select * from productss;
Empty set (0.00 sec)

mysql> select * from productss;
+-----+-----+-----+
| name                | price | ratings                |
+-----+-----+-----+
| Fire-Boltt Ninja Call | 1799  | 4.3 out of 5 stars |
+-----+-----+-----+
1 row in set (0.00 sec)

mysql>

```

Conclusion

The above Practical was studied and executed Successfully.

PRACTICAL-02

Aim:-Scrape an online Media Site for Data.

Objective

Use python to scrape all the news article Headlines and respective links to that news article from the Times Of India webpage.

Method

A powerful form of data collection, web scraping allows you to gather a wide array of information from websites and transfer them to spreadsheets and databases. Although it sounds difficult to the uninitiated, web scraping is actually quite simple if you use an Application Programming Interface (API). Using an API will automate a lot of the repetitive work associated with manual scraping — saving you a lot of time and energy that you can use for other tasks.

Program

```
import requests
from bs4 import BeautifulSoup

headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.3'}
#url = 'https://timesofindia.indiatimes.com/india&#39;
url = "https://timesofindia.indiatimes.com/india";

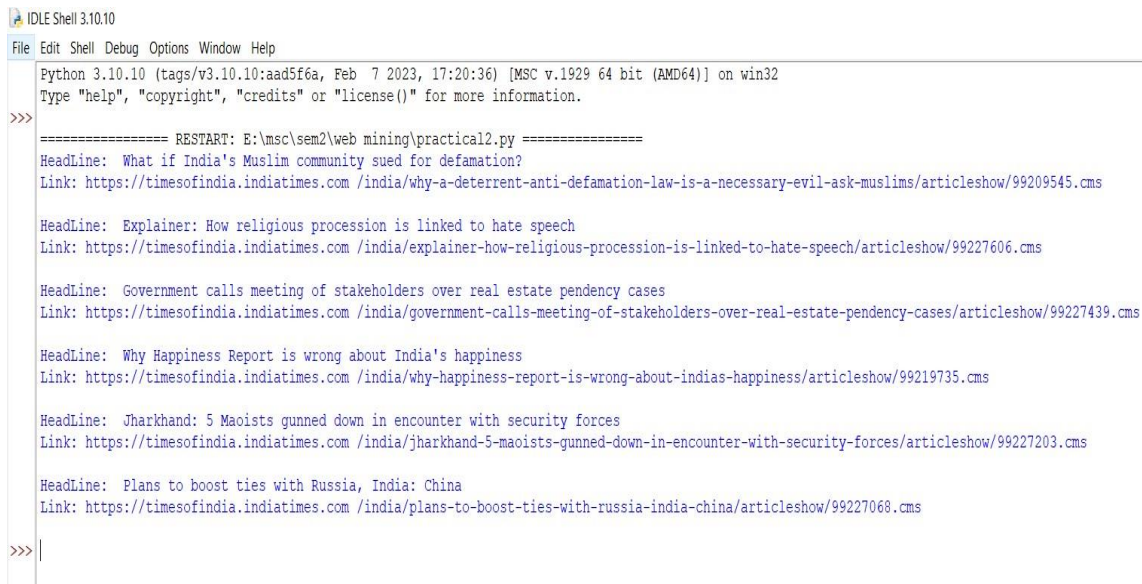
try:
    response = requests.get(url, headers=headers)
    soup = BeautifulSoup(response.text, 'html.parser')
    headlines = soup.find_all('span', {'class': "w_tle"})
    for headline in headlines[9:15]:
```

```

text = headline.text.strip()
link = headline.find('a')['href']
print("HeadLine: ",text)
print("Link: https://timesofindia.indiatimes.com",link)
print()
except Exception as e:
    print("Error:", e)

```

Output



```

IDLE Shell 3.10.10
File Edit Shell Debug Options Window Help
Python 3.10.10 (tags/v3.10.10:aad5f6a, Feb 7 2023, 17:20:36) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: E:\msc\sem2\web mining\practical2.py =====
HeadLine: What if India's Muslim community sued for defamation?
Link: https://timesofindia.indiatimes.com /india/why-a-deterrent-anti-defamation-law-is-a-necessary-evil-ask-muslims/articleshow/99209545.cms

HeadLine: Explainer: How religious procession is linked to hate speech
Link: https://timesofindia.indiatimes.com /india/explainer-how-religious-procession-is-linked-to-hate-speech/articleshow/99227606.cms

HeadLine: Government calls meeting of stakeholders over real estate pendency cases
Link: https://timesofindia.indiatimes.com /india/government-calls-meeting-of-stakeholders-over-real-estate-pendency-cases/articleshow/99227439.cms

HeadLine: Why Happiness Report is wrong about India's happiness
Link: https://timesofindia.indiatimes.com /india/why-happiness-report-is-wrong-about-indias-happiness/articleshow/99219735.cms

HeadLine: Jharkhand: 5 Maoists gunned down in encounter with security forces
Link: https://timesofindia.indiatimes.com /india/jharkhand-5-maoists-gunned-down-in-encounter-with-security-forces/articleshow/99227203.cms

HeadLine: Plans to boost ties with Russia, India: China
Link: https://timesofindia.indiatimes.com /india/plans-to-boost-ties-with-russia-india-china/articleshow/99227068.cms
>>>

```

Conclusion:-The above Practical was studied and executed successfully.

PRACTICAL-03

Aim:-Page Rank for link analysis using python Create a small set of pages namely page1, page2, page3 and page4 apply random walk on the same.

Objective

Use python to assign page rank to the data by the random walk.

Method

PageRank (PR) is an algorithm used by Google Search to rank websites in their search engine results. PageRank was named after Larry Page, one of the founders of Google. PageRank is a way of measuring the importance of website pages.

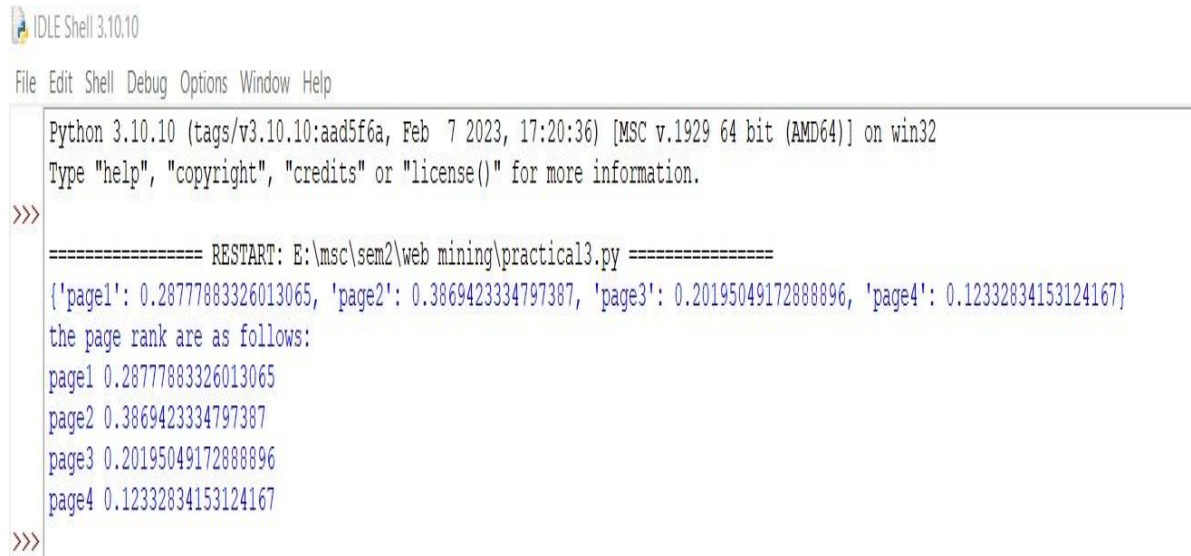
PageRank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites.

Program

```
import networkx as nx
G = nx.DiGraph()
G.add_nodes_from(['page1','page2','page3','page4'])
G.add_edges_from([('page1','page2'),('page1','page2'),('page2','page1'),('page2','page3'),('page3','p
age1'),('page3','page4'),('page4','page2')])
pr = nx.pagerank(G,alpha=0.85)
rw = nx.pagerank(G)
print(rw)
print('the page rank are as follows:')
print("page1",pr['page1'])
print("page2",pr['page2'])
print("page3",pr['page3'])
```

```
print("page4",pr['page4'])
```

Output



```

IDLE Shell 3.10.10
File Edit Shell Debug Options Window Help
Python 3.10.10 (tags/v3.10.10:aad5f6a, Feb  7 2023, 17:20:36) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: E:\msc\sem2\web mining\practical3.py =====
{'page1': 0.28777883326013065, 'page2': 0.3869423334797387, 'page3': 0.20195049172888896, 'page4': 0.12332834153124167}
the page rank are as follows:
page1 0.28777883326013065
page2 0.3869423334797387
page3 0.20195049172888896
page4 0.12332834153124167
>>>

```

Conclusion:-The above Practical was studied and executed Successfully.

PRACTICAL-04

Aim:-Perform Spam Classifier.

Objective

Use python to classify the given messages as spam or genuine

Method

Spam messages are messages sent to a large group of recipients without their prior consent, typically advertising for goods and services or business opportunities.

A spam message classification is a step towards building a tool for scam message identification and early scam detection.

Program

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB

# Load the spam dataset
spam_df = pd.read_csv('spam_data.csv')

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(spam_df['text'], spam_df['label'], test_size=0.2,
random_state=42)

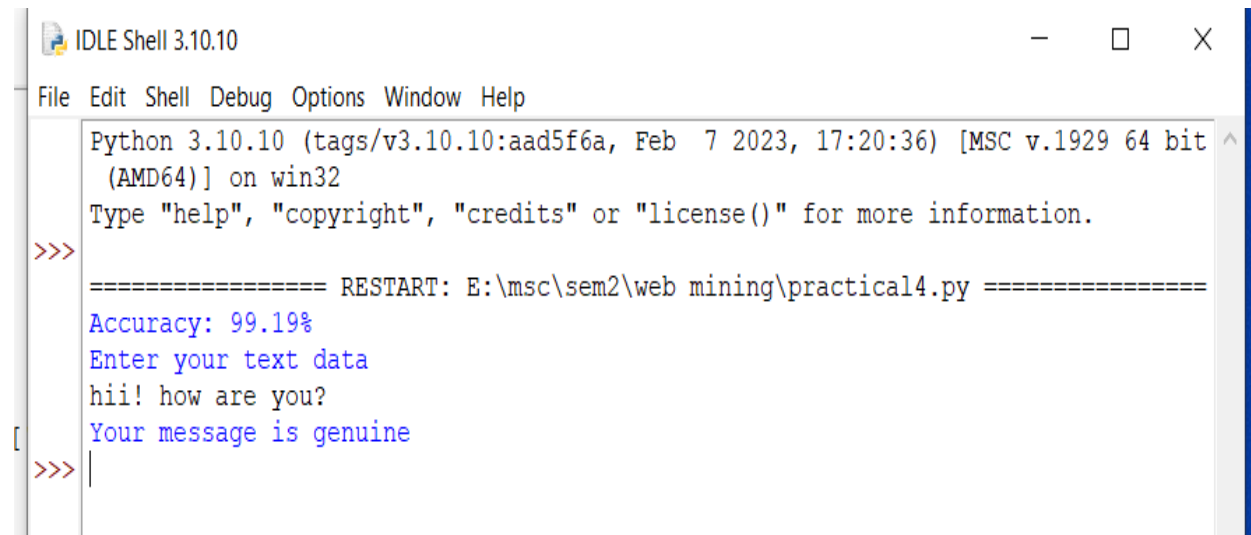
# Vectorize the text data using a Bag of Words model
vectorizer = CountVectorizer()
```

```
X_train_vect = vectorizer.fit_transform(X_train)
X_test_vect = vectorizer.transform(X_test)

# Train a Multinomial Naive Bayes classifier
clf = MultinomialNB()
clf.fit(X_train_vect, y_train)

# Evaluate the accuracy of the model on the test set
accuracy = clf.score(X_test_vect, y_test)
print("Accuracy: {:.2f}%".format(accuracy*100))

# Test the model on a new message
#You have won a prize! Click here to claim it now!
new_message = [input("Enter your text data\n")]
new_message_vect = vectorizer.transform(new_message)
prediction = clf.predict(new_message_vect)
if(prediction[0]=="spam"):
    print("Your message is a spam")
else:
    print("Your message is genuine")
```

Output:-

```
IDLE Shell 3.10.10
File Edit Shell Debug Options Window Help
Python 3.10.10 (tags/v3.10.10:aad5f6a, Feb  7 2023, 17:20:36) [MSC v.1929 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: E:\msc\sem2\web mining\practical4.py =====
Accuracy: 99.19%
Enter your text data
hii! how are you?
Your message is genuine
[
>>> |
```

Conclusion:-The above Practical was studied and executed Successfully.

PRACTICAL-05

Aim:-Demonstrate Text Mining and Web Page Preprocessing using meta information from the web pages (Local/Online).

Objective

It is used to examine and explore unstructured data for taking out meaningful patterns and textual data sources.

Method

Text mining is used for mining the data from a unstructured data and preprocessing is used to help to clean the records and determine the interesting user pattern.

Program

```
from bs4 import BeautifulSoup
import requests
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer

# Download stopwords and initialize stemmer
stop_words = set(stopwords.words('english'))
stemmer = PorterStemmer()

# Retrieve webpage content
url = 'https://www.britannica.com/'
response = requests.get(url)
```

```

# Parse HTML and extract meta tags
soup = BeautifulSoup(response.content, 'html.parser')
meta_tags = soup.find_all('meta')

# Extract meta information
title = ""
description = ""
keywords = []

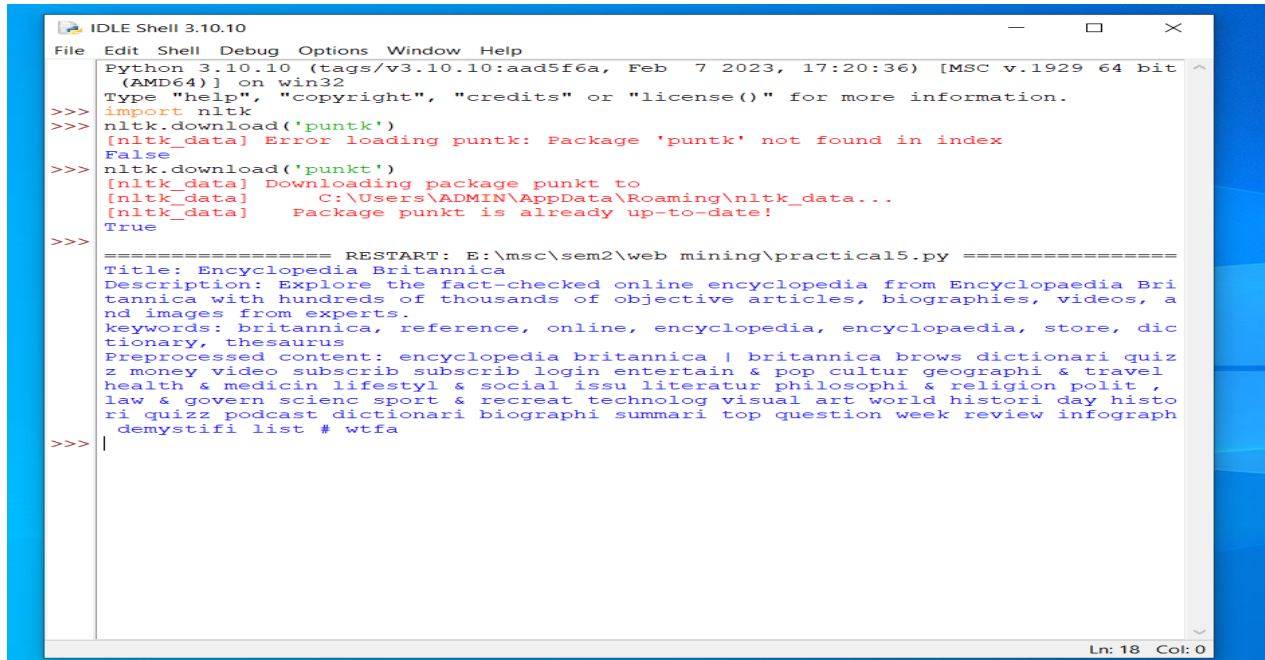
for tag in meta_tags:
    if tag.get('property')=='og:title':
        title=tag.get('content')
    if tag.get('property')=='og:description':
        description=tag.get('content')
    if tag.get('name')=='keywords':
        keywords=tag.get('content')

# Preprocess content
content = soup.get_text()
tokens = word_tokenize(content)
filtered_tokens = [token.lower() for token in tokens if token.lower() not in stop_words]
stemmed_tokens = [stemmer.stem(token) for token in filtered_tokens]
preprocessed_content = ' '.join(stemmed_tokens)

# Print results
print("Title:", title)
print('Description:', description)
print('keywords:', keywords)
print('Preprocessed content:', preprocessed_content[0:400])

```

Output



```

IDLE Shell 3.10.10
File Edit Shell Debug Options Window Help
Python 3.10.10 (tags/v3.10.10:aad5f6a, Feb 7 2023, 17:20:36) [MSC v.1929 64 bit
(AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import nltk
>>> nltk.download('punkt')
[nltk_data] Error loading punkt: Package 'punkt' not found in index
False
>>> nltk.download('punkt')
[nltk_data] Downloading package punkt to
[nltk_data] C:\Users\ADMIN\AppData\Roaming\nltk_data...
[nltk_data] Package punkt is already up-to-date!
True
>>>
===== RESTART: E:\msc\sem2\web mining\practical5.py =====
Title: Encyclopedia Britannica
Description: Explore the fact-checked online encyclopedia from Encyclopaedia Bri
tannica with hundreds of thousands of objective articles, biographies, videos, a
nd images from experts.
keywords: britannica, reference, online, encyclopedia, encyclopaedia, store, dic
tionary, thesaurus
Preprocessed content: encyclopedia britannica | britannica brows dictionari quiz
z money video subscrib subscrib login entertain & pop cultur geographi & travel
health & medicin lifestyl & social issu literatur philosophi & religion polit ,
law & govern scienc sport & recreat technolog visual art world histori day histo
ri quizz podcast dictionari biographi summari top question week review infograph
demystifi list # wtfa
>>>
Ln: 18 Col: 0

```

Conclusion:-The above Practical was studied and Executed Successfully.

PRACTICAL-06

Aim:- Apriori Algorithm implementation in case study.

Objective

The objective of the Apriori algorithm is to discover meaningful relationships and patterns in data.

Method

The Apriori algorithm is a popular algorithm for frequent item set mining and association rule learning in data mining and machine learning. The algorithm is designed to extract frequent item sets from a dataset and use these frequent item sets to generate association rules between items.

Code

```
import pandas as pd
import mlxtend
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori

# Example dataset
dataset = [['bread', 'milk', 'eggs'],
           ['bread', 'diapers', 'beer', 'eggs'],
           ['milk', 'diapers', 'beer', 'cola'],
           ['bread', 'milk', 'diapers', 'beer', 'cola'],
           ['bread', 'milk', 'diapers', 'beer']]

# Convert the dataset to a one-hot encoded matrix
te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
```

```

df = pd.DataFrame(te_ary, columns=te.columns_)

# Apply the Apriori algorithm to generate frequent itemsets
frequent_itemsets = apriori(df, min_support=0.6, use_colnames=True)

# Print the frequent itemsets
print(frequent_itemsets)

```

Output

```

= RESTART: D:/aakif/AAKIF/MS-C-I/Sem-2/Computer Science/WebMini
_6.py
  support      itemsets
0      0.8      (beer)
1      0.8      (bread)
2      0.8    (diapers)
3      0.8      (milk)
4      0.6    (beer, bread)
5      0.8    (beer, diapers)
6      0.6    (milk, beer)
7      0.6    (diapers, bread)
8      0.6    (milk, bread)
9      0.6    (milk, diapers)
10     0.6 (beer, diapers, bread)
11     0.6 (milk, beer, diapers)

```

Conclusion:- The above Practical was studied and Executed Successfully.

PRACTICAL-07

Aim:-Develop a basic crawler for the web search for user defined keywords.

Objective

By using the basic crawler we search for user defined keyword that it is there on webpage or not.

Method

A web crawler, crawler or web spider, is a computer program that's used to search and automatically index website content and other information over the internet. These programs, or bots, are most commonly used to create entries for a search engine index.

Code

```
import requests
from bs4 import BeautifulSoup

def web_crawler(url, keywords):
    # Make a request to the URL
    response = requests.get(url)

    # Check if the request was successful
    if response.status_code == 200:
        # Parse the HTML content using BeautifulSoup
        soup = BeautifulSoup(response.content, 'html.parser')

        # Find all the text on the page
        page_text = soup.get_text()

        # Check if any of the keywords are present on the page
```

```

for keyword in keywords:
    if keyword.lower() in page_text.lower():
        print(f'{keyword} found on {url}')
    else:
        print(f'{keyword} not found on {url}')

else:
    print(f'Request failed with status code {response.status_code}')

# Example usage
url = input("Enter the URL to be searched") #e.g.https://timesofindia.indiatimes.com/
keywords = []
print("Enter the keywords to be searched");
while(True):
    k=input("Enter the keyword")
    keywords.append(k)
    x=int(input("Enter 1 to give more keyword, enter 0 to exit"))
    if x==0:
        break
web_crawler(url, keywords)

```

Output

```
>>> = RESTART: D:/aakif/AAKIF/MSC-I/Sem-2/Computer Science/WebMining/Practical/practical_7.py
Enter the URL to be searchedhttps://timesofindia.indiatimes.com/
Enter the keywords to be searched
Enter the keywordkejriwal
Enter 1 to give more keyword, enter 0 to exit1
Enter the keywordPune Pllice
Enter 1 to give more keyword, enter 0 to exit1
Enter the keywordCCTV
Enter 1 to give more keyword, enter 0 to exit1
Enter the keywordElection
Enter 1 to give more keyword, enter 0 to exit0
kejriwal found on https://timesofindia.indiatimes.com/
Pune Pllice not found on https://timesofindia.indiatimes.com/
CCTV found on https://timesofindia.indiatimes.com/
Election found on https://timesofindia.indiatimes.com/
>>>
```

Conclusion:-The above Practical was studied and executed successfully.

PRACTICAL-08

Aim:- Develop a focused crawler for local search.

Objective

A focused crawler is a web crawler that collects Web pages that satisfy some specific property, by carefully prioritizing the crawl frontier and managing the hyperlink exploration process.

Method

A focused crawler **focuses its indexing on topic-relevant and particularly current websites.**

Code

```
import requests

def local_crawler(location, keywords):
    # Set up the API request URL
    url =
f'https://nominatim.openstreetmap.org/search.php?q={"+"}.join(keywords)}&format=json&address
details=1&limit=10&viewbox=&bounded=1&countrycodes='

    # Make the API request
    response = requests.get(url)

    # Check if the request was successful
    if response.status_code == 200:
        # Extract the results from the JSON response
        results = response.json()

        # Print out the name and address of each result
        for result in results:
```

```

name = result.get('display_name', 'Unknown')
address = ', '.join(filter(None, [result.get('address', {}).get('road'), result.get('address',
{}).get('city'), result.get('address', {}).get('state')]))
print(f'{name}: {address}\n')
else:
    print(f'Request failed with status code {response.status_code}')

```

Example usage

location = 'Thane, India' # City and state abbreviation

keywords = ['Hospital','Doctor','Thane']

local_crawler(location, keywords)

OutPut

```

= RESTART: D:/aakif/AAKIF/MSC-I/Sem-2/Computer Science/WebMining/Practical/Prac
tical_8.py
Doctor's Clinic, Seawoods NRI Service Road, Seawoods West, Navi Mumbai, Thane,
Maharashtra, 400706, India: Seawoods NRI Service Road, Navi Mumbai, Maharashtra
>>> |

```

Conclusion:-The above Practical was studied and executed successfully.

PRACTICAL-09

Aim:-Develop a program for deep search implementation to detect plagiarism in documents online.

Objective

To ensure that you didn't forget to cite anything, then you should use a plagiarism checker yourself. A plagiarism checker works by using advanced database software to scan for matches between your text and existing texts.

Method

Plagiarism can be detected by your professor or readers if the tone, formatting, or style of your text is different in different parts of your paper, or if they're familiar with the plagiarized source.

Code

```
# Import Required Libraries
import requests
from bs4 import BeautifulSoup
import difflib
import nltk

#nltk.download('stopwords') #run in the python idle
#nltk.download('punkt')

#run in the python idle
# Collect Data from User
text = input("Enter Text to Check for Plagiarism: ")

# Text Preprocessing

stop_words = set(nltk.corpus.stopwords.words('english'))
```

```

tokens = nltk.word_tokenize(text)
tokens = [token.lower() for token in tokens if token.isalpha()]
tokens = [token for token in tokens if token not in stop_words]

# Scrape Web for Similar Text
url = 'https://en.wikipedia.org/wiki/Java_%28programming_language%29'
response = requests.get(url)
soup = BeautifulSoup(response.content, 'html.parser')
soup_text = soup.get_text()
soup_tokens = nltk.word_tokenize(soup_text)
soup_tokens = [token.lower() for token in soup_tokens if token.isalpha()]
soup_tokens = [token for token in soup_tokens if token not in stop_words]
# Calculate Similarity
similarity = difflib.SequenceMatcher(None, tokens, soup_tokens).ratio()

# Set Threshold for Plagiarism
threshold = 0.002
print(similarity)
# Plagiarism Detection
if similarity >= threshold:
    print("Text is Plagiarized.")
else:
    print("Text is Not Plagiarized.")

```

Output

```
ical_9.py
Enter Text to Check for Plagiarism: Java is a high-level, class-based, object-or
iented programming language that is designed to have as few implementation depen
dencies as possible. It is a general-purpose programming language intended to le
t programmers write once, run anywhere (WORA), [17] meaning that compiled Java co
de can run on all platforms that support Java without the need to recompile.
0.010342846198046351
Text is Plagiarized.
```

Conclusion:-The above Practical was studied and executed successfully.

Practical-10

Aim:- Sentiment analysis for reviews by customers and visualize the same.

Objective

The aim of the study is to classify customer reviews into positive or negative sentiment. To analyze the association between customer reviews concerning different customers and visualize the same..

Method

Sentiment analysis is the process of analyzing digital text to determine if the emotional tone of the message is positive, negative, or neutral.

Code

```
# Import necessary libraries
#pip install textblob
#pip install vaderSentiment
import pandas as pd
import matplotlib.pyplot as plt
from textblob import TextBlob
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

# Load customer reviews into a dataframe
reviews_df = pd.read_csv('customer_reviews.csv')

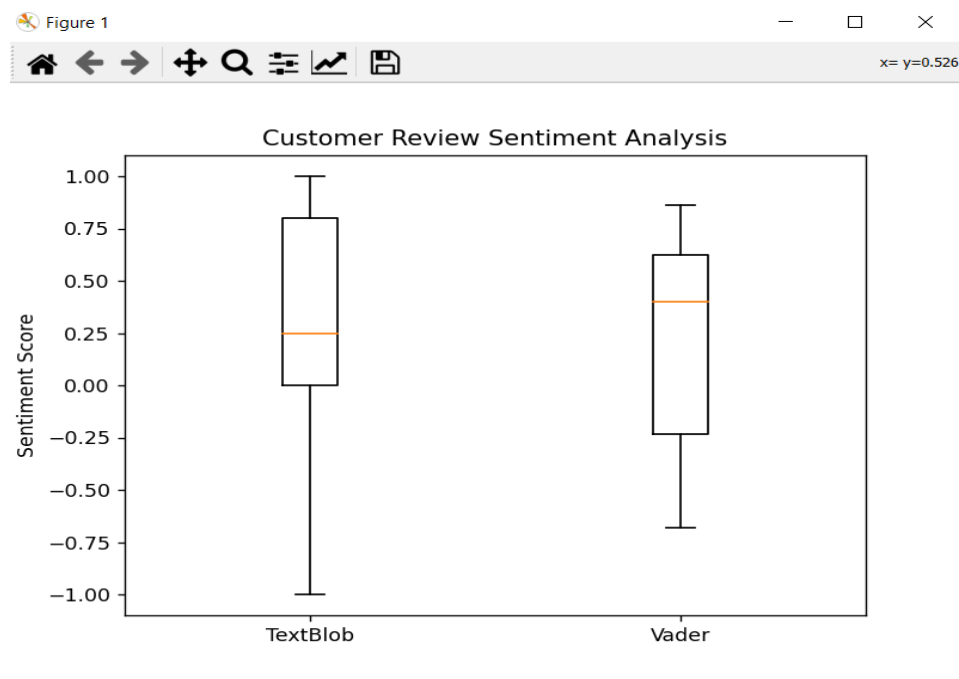
# Define functions for sentiment analysis using TextBlob and VaderSentiment
def get_textblob_sentiment(review):
    return TextBlob(review).sentiment.polarity

def get_vader_sentiment(review):
    analyzer = SentimentIntensityAnalyzer()
    return analyzer.polarity_scores(review)['compound']
```

```
# Apply sentiment analysis functions to customer reviews dataframe
reviews_df['TextBlob_Sentiment'] = reviews_df['review_text'].apply(get_textblob_sentiment)
reviews_df['Vader_Sentiment'] = reviews_df['review_text'].apply(get_vader_sentiment)

# Visualize sentiment analysis results using a box plot
plt.boxplot([reviews_df['TextBlob_Sentiment'], reviews_df['Vader_Sentiment']])
plt.xticks([1, 2], ['TextBlob', 'Vader'])
plt.ylabel('Sentiment Score')
plt.title('Customer Review Sentiment Analysis')
plt.show()
```

Output



Conclusion:-The above Practical was studied and executed successfully.