Text

Description automatically generated with medium confidence

**S.I.E.S College of Arts, Science and Commerce (Autonomous) Sion(W), Mumbai – 400 022.**

**CERTIFICATE**

**This is to certify that Mr. / Miss. \_\_\_\_\_\_**

**Roll No. \_\_\_\_\_\_\_\_\_ has successfully completed the necessary course of**

**experiments in subject of \_\_Information Retrival\_\_ during the**

**academic year 2022 – 2023 complying with the requirements for the course of T.Y.BSc Computer Science [Semester-VI ]**

**Prof. In-Charge Examination Date:**

**Shivani Deopa**

**Examiner’s Signature & Date:**

**College Seal**

**& Date**

INDEX

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sr No | Aim | Page No | Date | Sign |
| 1 | Write a program to demonstrate bitwise operation. | 3 | 24/11/22 |  |
| 2 | Implement Page Rank Algorithm. | 5 | 17/12/22 |  |
| 3 | Implement Dynamic programming algorithm for computing the edit distance between strings s1 and s2. | 7 | 07/01/23 |  |
| 4 | Write a program to Compute Similarity between two text documents. | 8 | 21/01/23 |  |
| 5 | Write a program for Pre-processing of a Text Document: stop word removal | 11 | 28/01/23 |  |
| 6 | Write a program for mining Twitter to identify tweets for a specific period and identify trends and named entities. | 12 | 04/02/23 |  |
| 7 | Write a program to implement simple web crawler. | 14 | 11/02/23 |  |
| 8 | Write a program to parse XML text and save it to CSV | 15 | 25/02/23 |  |

**Practical 1**

**AIM:**

Write a program to demonstrate bitwise operation.

**CODE:**

list1=[]

list2=[]

def intersection(list1,list2):

p1=0

p2=0

answer=[]

while p1!=len(list1) and p2 != len(list2):

if list1[p1]==list2[p2]:

answer.append(list1[p1])

p1+=1

p2+=1

elif list1[p1]<list2[p2]:

p1+=1

else :

p2+=1

return(answer)

n = int(input("Enter number of elements for list 1: "))

for i in range(0, n):

ele = int(input())

list1.append(ele)

n2 = int(input("Enter number of elements for list 2: "))

for i in range(0, n2):

ele = int(input())

list2.append(ele)

print("intersection of the given two list are ",intersection(list1,list2))

list1=[1,2,4,11,31,45,173]

list2=[2,31,54,101,173]

def intersection(list1,list2):

p1=0

p2=0

answer=[]

while p1!=len(list1) and p2 != len(list2):

if list1[p1]==list2[p2]:

answer.append(list1[p1])

p1+=1

p2+=1

elif list1[p1]<list2[p2]:

p1+=1

else :

p2+=1

return(answer)

print("intersection of the given two list are ",intersection(list1,list2))

**OUTPUT:**

Graphical user interface, text, application

Description automatically generated

**Practical 2**

**AIM:** Implement Page Rank Algorithm.

**CODE:**

package pageranking;

import java.util.Scanner;

public class PageRanking {

int path[][] = new int[10][10];

double pagerank[] = new double[10];

public void calc(double totalNodes) {

double InitialPageRank;

double OutgoingLinks = 0; /\* C(Wi) \*/

double DumpingFactor = 0.85;

double TempPageRank[] = new double[10];

int ExternalNodeNumber;

int InternalNodeNumber;

int k = 1;

int ITERATION\_STEP = 1;

InitialPageRank = 1 / totalNodes;

System.out.println("Total Number of Nodes: " + totalNodes + "Initial PageRank of all Nodes: " + InitialPageRank);

for (k = 1; k <= totalNodes; k++) {

this.pagerank[k] = InitialPageRank;

}

System.out.println("Initial PageRank values, 0th step");

for (k = 1; k <= totalNodes; k++) {

System.out.println("Page Rank of " + k + " is: " + this.pagerank[k]);

}

while (ITERATION\_STEP <= 2) {

for (k = 1; k <= totalNodes; k++) {

TempPageRank[k] = this.pagerank[k];

this.pagerank[k] = 0;

}

for (InternalNodeNumber = 1; InternalNodeNumber <= totalNodes; InternalNodeNumber++) {

for (ExternalNodeNumber = 1; ExternalNodeNumber <= totalNodes; ExternalNodeNumber++) {

if (this.path[ExternalNodeNumber][InternalNodeNumber] == 1) {

k = 1;

OutgoingLinks = 0;

while (k <= totalNodes) {

if (this.path[ExternalNodeNumber][k] == 1) {

OutgoingLinks = OutgoingLinks + 1;

}

k = k + 1;

}

this.pagerank[InternalNodeNumber] += TempPageRank[ExternalNodeNumber] \* (1 / OutgoingLinks);

}

}

}

System.out.println("After " + ITERATION\_STEP + "th step");

for (k = 1; k <= totalNodes; k++) {

System.out.println("Page Rank of " + k + "in" + this.pagerank[k]);

ITERATION\_STEP = ITERATION\_STEP + 1;

}

for (k = 1; k <= totalNodes; k++) {

this.pagerank[k] = (1 - DumpingFactor) + DumpingFactor \* this.pagerank[k];

}

System.out.println("Final Page Rank: ");

for (k = 1; k <= totalNodes; k++) {

System.out.println("Page Rank of " + k + "is: " + this.pagerank[k]);

}

}

}

public static void main(String[] args) {

int nodes, i, j;

Scanner in = new Scanner(System.in);

System.out.println("Enter the Number of WebPages");

nodes = in.nextInt();

PageRanking p = new PageRanking();

System.out.println("Enter the Adjacency matrix with 1->PATH & 0->NO PATH Be"+"tween two Webpages:");

for(i=1;i<=nodes;i++){

for (j=1;j<=nodes;j++){

p.path[i][j]=in.nextInt();

if(j==1){

p.path[i][j]=0;

}

}

p.calc(nodes);

}

}

}

**OUTPUT:**

A picture containing text

Description automatically generated

**Practical 3**

**AIM:** Implement Dynamic programming algorithm for computing the edit distance between strings s1 and s2.

**CODE:**

def editDistance(s1,s2,m,n):

if m==0:

return n

if n==0:

return m

if(s1[m-1]==s2[n-1]):

return editDistance(s1,s2,m-1,n-1)

return 1+min(editDistance(s1,s2,m,n-1),editDistance(s1,s2,m-1,n-1))

s1="network"

s2="cats"

editDistance(s1,s2,len(s1),len(s2))

**OUTPUT:**

Text

Description automatically generated with medium confidence

**Practical 4**

**AIM:** Write a program to Compute Similarity between two text documents.

**CODE:**

package pageranking;

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

//Aim: Program to find similarities between two documents

public class PageSimilarity {

public static void main(String[] args) throws IOException {

BufferedReader reader1 = new BufferedReader(new FileReader("

BufferedReader reader2 = new BufferedReader(new FileReader("

String line1 = reader1.readLine();

String line2 = reader2.readLine();

boolean areEqual = true;

int lineNum = 1;

while (line1 != null || line2 != null) {

if (line1 == null || line2 == null) {

areEqual = false;

break;

} else if (!line1.equalsIgnoreCase(line2)) {

areEqual = false;

break;

}

line1 = reader1.readLine();

line2 = reader2.readLine();

lineNum++;

}

if (areEqual) {

System.out.println("Two files have same content");

} else {

System.out.println("Two files have different content. They differ at line " + lineNum);

System.out.println("File1 has " + line1 + " and file2 has " + line2 + " at line " + lineNum);

}

reader1.close();

reader2.close();

}

}

**OUTPUT:**

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

**Practical 5**

**AIM:** Write a program for Pre-processing of a Text Document: stop word removal.

**CODE:**

import nltk

#natural language tool kit

nltk.download('stopwords')

nltk.download('punkt')

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

eg=input('Enter any sentence : ')

stop\_words=set(stopwords.words('english'))

tokens=word\_tokenize(eg)

filtered=[]

for w in tokens:

if w not in stop\_words:

filtered.append(w)

print(tokens)

print(filtered)

**OUTPUT:** Graphical user interface, text, application

Description automatically generated

**Practical 6**

**AIM:** Write a program for mining Twitter to identify tweets for a specific period and identify trends and named entities.

**CODE:**

import re

import tweepy

from tweepy import OAuthHandler

from textblob import TextBlob

class TwitterClient(object):

def \_\_init\_\_(self):

consumer\_key='LlnXUxe64nBJQs0vCFPTj6Vkx'

consumer\_secret='PyhmahYOAONBaDBdw5U2mnIXCw6XdGKza0kgw35Dl86du5EegD'

access\_token='4313447914-5qh3YuUbTegPvjTMhfmeQONudsQgBDVJtHuijHB'

access\_token\_secret='DBizwMGRuSemXtq3HbOaatBHa7rFXUywofhwYQY91Zj9N'

try :

self.auth=OAuthHandler(consumer\_key,consumer\_secret)

self.auth.set\_access\_token(access\_token,access\_token\_secret)

self.api=tweepy.API(self.auth)

except:

print("Error : Authentication Failed")

def clean\_tweet(self,tweet):

return ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\s+)"," ",tweet).split())

def get\_tweet\_sentiment(self,tweet):

analysis = TextBlob(self.clean\_tweet(tweet))

if analysis.sentiment.polarity>0:

return 'Positive'

else:

return 'Negative'

def get\_tweets(self,query,count=10):

tweets=[]

try:

fetched\_tweets=self.api.search\_tweets(q=query,count=count)

for tweet in fetched\_tweets:

parsed\_tweet={}

parsed\_tweet['text']=tweet.text

parsed\_tweet['sentiment']=self.get\_tweet\_sentiment(tweet.text)

if tweet.retweet\_count>0:

if parsed\_tweet not in tweets:

tweets.append(parsed\_tweet)

else:

tweets.append(parsed\_tweet)

return tweets

except exception as e:

print(e)

def main():

api=TwitterClient()

tweets=api.get\_tweets(query='Corona Virus',count=60)

ptweets=[tweet for tweet in tweets if tweet['sentiment']=='Positive']

print('Positive tweets percentange : {} %'.format(100\*len(ptweets)/len(tweets)))

ntweets=[tweet for tweet in tweets if tweet['sentiment']=='Negative']

print('Negative tweets percentange : {} %'.format(100\*len(ntweets)/len(tweets)))

print("\n\nPositive tweets:")

for tweet in ptweets[:5]:

print(tweet['text'])

print("\n\nNegativee tweets:")

for tweet in ntweets[:5]:

print(tweet['text'])

if \_\_name\_\_=="\_\_main\_\_":

main()

OUTPUT:

A screenshot of a computer

Description automatically generated

**Practical 7**

**AIM:** Write a program to implement simple web crawler.

**CODE:**

import requests

from parsel import Selector

import time

start =time.time()

response=requests.get("http://recurship.com/")

selector=Selector(response.text)

href\_links=selector.xpath('//a/@href').getall()

image\_links=selector.xpath('//img/@src').getall()

print('-------------------------------------Links----------------------------------')

print(href\_links)

print('------------------------------------Images----------------------------------')

print(image\_links)

end=time.time()

print('Time taken in seconds : ',end-start)

**OUTPUT:**

Text, letter

Description automatically generatedText, letter

Description automatically generated

**Practical 8**

**AIM:** Write a program to parse XML text and save it to CSV

**CODE:**

import csv

import requests

import xml.etree.ElementTree as ET

def loadRSS():

url="https://timesofindia.indiatimes.com/rssfeeds/1081479906.cms"

response=requests.get(url)

with open("topnewsfeed.xml",'wb') as f:

f.write(response.content)

def parseXML(xml\_file):

tree=ET.parse(xml\_file)

root=tree.getroot()

newsitems=[]

for item in root.findall('./channel/item'):

news={}

for child in item[:5]:

news[child.tag]=child.text.encode('utf8')

newsitems.append(news)

return newsitems

def savetoCSV(newsitems,filename):

fields=['guid','title','pubDate','description','link']

with open(filename ,'w') as csvfile:

writer=csv.DictWriter(csvfile,fieldnames=fields)

writer.writeheader()

writer.writerows(newsitems)

def main():

loadRSS()

newsitems=parseXML('topnewsfeed.xml')

savetoCSV(newsitems,'topnews.csv')

if \_\_name\_\_=="\_\_main\_\_":

main()

OUTPUT:

Graphical user interface, application, table, Excel

Description automatically generated