

R.D.&S.H.NationalCollege&S.W.A.ScienceCollege

Bandra, Mumbai**–** 400050.

Department of Computer Science

CERTIFICATE

This is to certify that Mr./Ms. AVINASH RAJKUMAR KAURAN of

COMPUTER SCIENCEclass (TYBSC CS Semester VI) has satisfactorily completed 05

Practicals, in the subject of INFORMATION RETRIEVAL

as a

part of B.Sc. Degree Course in Computer Science during the academic year

2021 – 2022.

Date of Certification:

Faculty Incharge Head,

Department Computer Science

Signature of Examiner

**INDEX**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SR.NO** | **DATE** | **AIM** | **PAGE NO** |  |
| **1** | **10/01/22** | Write a program to perform bitwise operations |  |  |
| **2** | **24/01/22** | Write a program to compute similarities between two documents. |  |  |
| **3** | **07/02/22** | Write a program to implement Levenshtein Distance or editDistance. |  |  |
| **4** | **21/01/22** | Write a program to implement Page Rank Algorithm. |  |  |
| **5** | **07/03/22** | Write a program to implement a Web Crawler. |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

PRACTICAL 1

Aim : Write a program to perform bitwise operations

a = int(input("Enter your value of a : "))

print(a)

b = int(input("Enter your value of b : "))

print(b)

c = 0;

c = a &b;

print("Line 1 - Value of c is ", c)

c = a | b;

print ("Line 2 - Value of c is ", c)

c = a ^ b;

print ("Line 3 - Value of c is ", c)

c = ~a;

print ("Line 4 - Value of c is ", c)

c = a <<2;

print ("Line 5 - Value of c is ", c)

c = a >>2;

print ("Line 6 - Value of c is ", c)

OUTPUT:

Text

Description automatically generated

PRACTICAL 2

Aim : Write a program to compute similarities between two documents.

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize

import numpy as np

import nltk

nltk.download("punkt")

nltk.download("stopwords")

def process(file):

raw=open (file).read()

tokens=word\_tokenize(raw)

words=[w.lower() for w in tokens]

porter= nltk.PorterStemmer()

stemmed\_tokens=[porter.stem(t) for t in words]

# removing stop words

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

filtered\_tokens=[w for w in stemmed\_tokens if not w instop\_words]

#count words

count=nltk.defaultdict(int)

for word in filtered\_tokens:

count[word]+=1

return count;

def cos\_sim(a,b):

dot\_product=np.dot(a,b)

norm\_a=np.linalg.norm(a)

norm\_b=np.linalg.norm(b)

return dot\_product/(norm\_a \* norm\_b)

def getSimilarity(dict1,dict2):

all\_words\_list=[]

for key in dict1:

all\_words\_list.append(key)

for key in dict2:

all\_words\_list.append(key)

all\_words\_list\_size=len(all\_words\_list)

v1=np.zeros(all\_words\_list\_size,dtype=np.int)

v2=np.zeros(all\_words\_list\_size,dtype=np.int)

i=0

for (key) in all\_words\_list:

v1[i]=dict1.get(key,0)

v2[i]=dict2.get(key,0)

i=i+1

return cos\_sim(v1,v2)

if \_\_name\_\_ == '\_\_main\_\_':

dict1=process("text1.txt")

dict2=process("text2.txt")

print("Similarity between two text documents",getSimilarity(dict1,dict2))

Text

Description automatically generated

PRACTICAL 3

Aim : Write a program to implement Levenshtein Distance or editDistance.

Given two strings str1 and str2 and below operations that can performed on str1.

Find minimum number of edits (operations) required to convert ‘str1’ into ‘str2’.

Below are the three operations of editDistance

1) Insert

2) Remove

3) Replace

For example in our code we will be converting string "sunday" to "saturday"

Directly by watching we can say that only 2 operations will be needed to convert "sun" to "sat"

That is by using the operation "Replace"

s = s (No operation needed)

u = a (Relplaced u with a) means (one operation)

n = t (Replaced n with t) means (one operation)

In total 2 operations needed.

Execute the code edirDistance.py

def editDistance(str1, str2, m , n):

# If first string is empty, the only option is to

# insert all characters of second string into first

if m==0:

return n

# If second string is empty, the only option is to

# remove all characters of first string

if n==0:

return m

# If last characters of two strings are same, nothing

# much to do. Ignore last characters and get count for

# remaining strings.

if str1[m-1]==str2[n-1]:

return editDistance(str1,str2,m-1,n-1)

# If last characters are not same, consider all three

# operations on last character of first string, recursively

# compute minimum cost for all three operations and take

# minimum of three values.

return 1 + min(editDistance(str1, str2, m, n-1), # Insert

editDistance(str1, str2, m-1, n), # Remove

editDistance(str1, str2, m-1, n-1) # Replace

)

# Driver program to test the above function

str1 = "sun"

str2 = "sat"

print("The total number of operations needed is : ",editDistance(str1, str2, len(str1), len(str2)))

Text

Description automatically generated with medium confidence

PRACTICAL 4

Aim : Write a program to implement Page Rank Algorithm.

import numpy as np

from scipy.sparse import csc\_matrix

from fractions import Fraction

def float\_format(vector, decimal):

return np.round((vector).astype(np.float), decimals=decimal)

G = np.matrix([[1,1,0],

[1,0,1],

[0,1,0]])

n=len(G)

#print(n)

# transform G into markov matrix A

M = csc\_matrix(G,dtype=np.float)

rsums = np.array(M.sum(1))[:,0]

ri, ci = M.nonzero()

M.data /= rsums[ri]

# WWW matrix

# we have 3 webpages and probability of landing to each one is 1/3

#(default Probability)

#n=len(M)

dp = Fraction(1,n)

E = np.zeros((3,3))

E[:] = dp

# taxation

beta = 0.85

# WWW matrix

A = beta \* M + ((1-beta) \* E)

# initial vector

r = np.matrix([dp, dp, dp])

r = np.transpose(r)

previous\_r = r

for it in range(1,30):

r = A \* r

#check if converged

if (previous\_r==r).all():

break

previous\_r = r

print ("Final:\n", float\_format(r,3))

print( "sum", np.sum(r))

Text

Description automatically generated

PRACTICAL 5

Aim : Write a program to implement a Web Crawler.

from html.parser import HTMLParser

from urllib.request import urlopen

from urllib import parse

import sys, json

class LinkParser(HTMLParser):

def handle\_starttag(self, tag, attrs):

if tag == "a":

for (key, value) in attrs:

if key == "href":

newUrl = parse.urljoin(self.baseUrl, value)

self.links = self.links + [newUrl]

def getLinks(self, url):

self.links = []

self.baseUrl = url

response = urlopen(url)

if "text/html" in response.getheader("Content-Type"):

htmlContent = response.read()

htmlString = htmlContent.decode("utf-8")

self.feed(htmlString)

response.close()

return htmlString, self.links

else:

return "", []

def crawl(url, word):

# List of found urls

foundUrl = []

# List of already visited url to prevent revisiting the same url twice

visitedURL = []

# Keeping count of all the pages visited

numberVisited = 0;

# If no words found show error

foundWord = False

# Starting the parser class

parser = LinkParser()

# Checking the first url

data, links = parser.getLinks(url)

links.append(url)

# Looping all the links

for link in links:

# Kinda straight foward...

numberVisited = numberVisited + 1

try:

# Checking if link has not been visited yet

if link not in visitedURL:

# Appending link to VisiterURL list

visitedURL.append(link)

data, li = parser.getLinks(link)

print (numberVisited, "Scanning URL ", link)

if data.find(word) > -1:

foundWord = True

foundUrl.append(link)

print("-" \* 10)

print(" ")

print("The word", word, "was found at", link)

print(" ")

print("-" \* 10)

else:

print ("Matches Not Found")

except:

print (" \*\*Failed \*\*", "")

#If the word was never found show the error

if foundWord == False:

print ("The word", word, "was not found!")

print ("Finished, crawled", numberVisited, "pages")

print (json\_list(foundUrl))

def json\_list(list):

lst = []

d = {}

for pn in list:

d=pn

lst.append(d)

return json.dumps(lst, separators=(',',':'))

crawl("https://www.facebook.com", "login");

\*\*Failed \*\*

Finished, crawled 48 pages

["https://www.facebook.com","https://www.facebook.com/recover/initiate/?privacy\_mutation\_token=eyJ0eXBlIjowLCJjcmVhdGlvbl90aW1lIjoxNjQ5NjU5ODc1LCJjYWxsc2l0ZV9pZCI6MzgxMjI5MDc5NTc1OTQ2fQ%3D%3D&ars=facebook\_login","https://es-la.facebook.com/","https://fr-fr.facebook.com/","https://zh-cn.facebook.com/","https://ar-ar.facebook.com/","https://pt-br.facebook.com/","https://it-it.facebook.com/","https://ko-kr.facebook.com/","https://de-de.facebook.com/","https://hi-in.facebook.com/","https://ja-jp.facebook.com/","https://www.facebook.com/reg/","https://www.facebook.com/login/","https://messenger.com/","https://www.facebook.com/lite/","https://www.facebook.com/watch/","https://www.facebook.com/places/","https://www.facebook.com/games/","https://www.facebook.com/marketplace/","https://pay.facebook.com/","https://www.oculus.com/","https://portal.facebook.com/","https://www.bulletin.com/","https://www.facebook.com/local/lists/245019872666104/","https://www.facebook.com/fundraisers/","https://www.facebook.com/biz/directory/","https://www.facebook.com/votinginformationcenter/?entry\_point=c2l0ZQ%3D%3D","https://www.facebook.com/groups/explore/","https://about.facebook.com/","https://www.facebook.com/ad\_campaign/landing.php?placement=pflo&campaign\_id=402047449186&nav\_source=unknown&extra\_1=auto","https://developers.facebook.com/?ref=pf","https://www.facebook.com/careers/?ref=pf","https://www.facebook.com/privacy/explanation/","https://www.facebook.com/policies/cookies/","https://www.facebook.com/help/568137493302217","https://www.facebook.com/policies?ref=pf","https://www.facebook.com/help/?ref=pf","https://www.facebook.com/settings"]

