# PRACTICAL NO. 1

## Aim: Write a program to demonstrate bitwise operation

**Theory**

Bitwise operators operate on one or more bit pattern or binary numbers at the level of individual bits.

* Bitwise AND: Text two equal length binary represent to and perform logical AND operation on each pair of the bit. If both bits in compare operation r, the result is one(1\*1=1)
* Bitwise OR(1): A bitwise OR text two bits of equal length and perform logical OR(1) operations on each pair of bits.If both bits are zero then result is zero.
* Bitwise NOT(1): A bitwise NOT is unary operation that perform logical negotiation on each bit.
* sklearn.feature\_extraction.text

It conevrts a collection of text document to a matrix of taken counts.

* CountVectorizer():

This provide a simple way to both tokenize- a collection text document and known word.

* Fit.transform():

This method applied feature extraction objects such as CountVectorizer that determines what features it will base feature transformation on.

## Program:

import pandas as pd

from sklearn.feature\_extraction.text import CountVectorizer corpus=[

'this is the first document.',

'this document is second document.', 'and this is the third one.',

'is this the first document?',

]

vectorizer= CountVectorizer() X=vectorizer.fit\_transform(corpus) print("fit transform is ") print(X.toarray())

df=pd.DataFrame(X.toarray(),columns=vectorizer.get\_feature\_names()) print("the generated data frame is")

print(df)

alldata= df[(df['this']==1)&(df['first']==1)]

print("indices where 'this'and 'first'terms are present are ",alldata.index.tolist()) ordata= df[(df['this']==1)|(df['first']==1)]

print("indices where either of 'this'and 'first'terms are present are ",ordata.index.tolist())

notdata=df[(df['and']!=1)]

print("indices where 'and' term is not present ",notdata.index.tolist

# PRACTICAL NO 2

## Aim: Implement PageRank Algorithm.

**Theory**

The PageRank Algorithm:

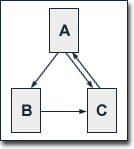
The original PageRank algorithm was described by Lawrence Page and Sergey Brin in several publications. It is given by

PR(A) = (1-d) + d (PR(T1)/C(T1) + ... + PR(Tn)/C(Tn))

http://pr.efactory.de/i/dot.gifhttp://pr.efactory.de/i/dot.gifwhere

|  |
| --- |
| PR(A) is the PageRank of page A, |
| PR(Ti) is the PageRank of pages Ti which link to page A, |
| C(Ti) is the number of outbound links on page Ti and |
| d is a damping factor which can be set between 0 and 1. |

http://pr.efactory.de/i/dot.gifhttp://pr.efactory.de/i/dot.gifExample:

.

A small web consisting of three pages A, B and C, whereby page A links to the pages B and C, page B links to page C and page C links to page A. the damping factor d is usually set to 0.85, but to keep the calculation simple we set it to 0.5. The exact value of the damping factor d admittedly has effects on PageRank, but it does not influence the fundamental principles of PageRank. So, we get the following equations for the PageRank calculation:

PR(A) = 0.5 + 0.5 PR(C)

PR(B) = 0.5 + 0.5 (PR(A) / 2)

PR(C) = 0.5 + 0.5 (PR(A) / 2 + PR(B))

These equations can easily be solved. We get the following PageRank values for the single pages:

PR(A) = 14/13 = 1.07692308 PR(B) = 10/13 = 0.76923077 PR(C) = 15/13 = 1.15384615

It is obvious that the sum of all pages' PageRanks is 3 and thus equals the total number of web pages. As shown above this is not a specific result for our simple example.

For our simple three-page example it is easy to solve the according equation system to determine PageRank values. In practice, the web consists of billions of documents and it is not possible to find a solution by inspection.

The Iterative Computation of PageRank

Because of the size of the actual web, the Google search engine uses an approximative, iterative computation of PageRank values. This means that each page is assigned an initial starting value and the PageRanks of all pages are then calculated in several computation circles based on the equations determined by the PageRank algorithm. The iterative calculation shall again be illustrated by our three-page example, whereby each page is assigned a starting PageRank value of 1.

|  |  |  |  |
| --- | --- | --- | --- |
| Iteration | PR(A) | PR(B) | PR(C) |
| 0 | 1 | 1 | 1 |
| 1 | 1 | 0.75 | 1.125 |
| 2 | 1.0625 | 0.765625 | 1.1484375 |
| 3 | 1.07421875 | 0.76855469 | 1.15283203 |
| 4 | 1.07641602 | 0.76910400 | 1.15365601 |
| 5 | 1.07682800 | 0.76920700 | 1.15381050 |
| 6 | 1.07690525 | 0.76922631 | 1.15383947 |
| 7 | 1.07691973 | 0.76922993 | 1.15384490 |
| 8 | 1.07692245 | 0.76923061 | 1.15384592 |
| 9 | 1.07692296 | 0.76923074 | 1.15384611 |
| 10 | 1.07692305 | 0.76923076 | 1.15384615 |
| 11 | 1.07692307 | 0.76923077 | 1.15384615 |
| 12 | 1.07692308 | 0.76923077 | 1.15384615 |

We see that we get a good approximation of the real PageRank values after only a few iterations. According to publications of Lawrence Page and Sergey Brin, about 100 iterations are necessary to get a good approximation of the PageRank values of the whole web.

## Program:

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) M=csc\_matrix(G,dtype=np.float) rsums=np.array(M.sum(1))[:,0] ri,ci=M.nonzero() M.data/rsums[ri] dp=Fraction(1,n) E=np.zeros((3,3))

E[:]=dp

beta=0.85 A=beta\*M+((1-beta)\*E) r=np.matrix([dp,dp,dp]) r=np.transpose(r) previous\_r=r

for it in range(1,30): r=A\*r if(previous\_r==r).all():

break previous\_r=r

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

## Output:

runfile('C:/Users/ckt/prac2ir.py', wdir='C:/Users/ckt') 3

('Final:\n', array([[0.617], [0.617],

[0.333]]))

('Final:\n', array([[1.127], [0.886],

[0.603]]))

('Final:\n', array([[1.841], [1.601],

[0.884]]))

('Final:\n', array([[3.142], [2.533],

[1.577]]))

('Final:\n', array([[5.186], [4.373],

[2.515]]))

('Final:\n', array([[8.729], [7.15 ],

[4.321]]))

('Final:\n', array([[14.507], [12.103],

[ 7.087]]))

('Final:\n', array([[24.303], [20.04 ],

[11.972]]))

('Final:\n', array([[40.507], [33.65 ],

[19.85 ]]))

('Final:\n', array([[67.734], [56.004],

[33.303]]))

('Final:\n', array([[113.029], [ 93.733],

[ 55.455]]))

('Final:\n', array([[188.859], [156.323],

[ 92.784]]))

('Final:\n', array([[315.303], [261.295],

[154.773]]))

('Final:\n', array([[526.677], [436.133],

[258.669]]))

('Final:\n', array([[879.463], [728.619],

[431.787]]))

('Final:\n', array([[1468.863], [1216.556],

[ 721.319]]))

('Final:\n', array([[2452.943], [2031.992],

[1204.41 ]]))

('Final:\n', array([[4096.662], [3393.217],

[2011.66 ]]))

('Final:\n', array([[6841.474], [5667.15 ],

[3359.311]]))

('Final:\n', array([[11425.727], [ 9464.064],

[ 5610.475]]))

('Final:\n', array([[19081.335], [15805.785],

[ 9369.467]]))

('Final:\n', array([[31866.881], [26396.012],

[15647.746]]))

('Final:\n', array([[53218.991], [44082.966],

[26132.142]]))

('Final:\n', array([[88878.368], [73620.168],

[43642.226]]))

('Final:\n', array([[148430.794], [122949.543],

[ 72884.181]]))

('Final:\n', array([[247886.512], [205330.955],

[121720.337]]))

('Final:\n', array([[413981.737], [342912.712],

[203278.202]]))

('Final:\n', array([[691368.915], [572679.581],

[339484.438]]))

('Final:\n', array([[1154617.868], [ 956401.996],

[ 566954.29 ]]))

# PRACTICAL NO 3

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

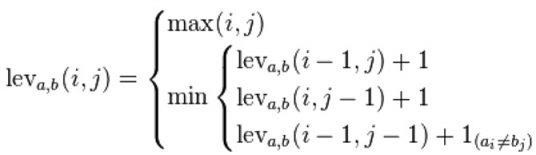
**Theory**

The Levenshtein distance is a string metric for measuring difference between two sequences. Informally, the Levenshtein distance between two words is the minimum number of single

character edits (i.e. insertions, deletions or substitutions) required to change one word into the other. It is named after [Vladimir Levenshtein,](https://en.wikipedia.org/wiki/Vladimir_Levenshtein) who considered this distance in 1965.

Levenshtein distance may also be referred to as edit distance, although it may also denote a larger family of distance metrics. It is closely related to pairwise string alignments.

Definition

Mathematically, the Levenshtein distance between two strings a, b (of length |a| and |b| respectively) is given by leva,b(|a|,|b|) where:

where 1(ai≠bi) is the indicator function equal to 0 when ai≠bi and equal to 1 otherwise, and leva, b(i,j) is the distance between the first i characters of a and the first j characters of b.

Note that the first element in the minimum corresponds to deletion (from a to b), the second to insertion and the third to match or mismatch, depending on whether the respective symbols are the same.

## Program:

import numpy as np def levenshelin(s1,s2):

size\_x=len(s1)+1 size\_y=len(s2)+1 matrix=np.zeros((size\_x,size\_y)) for x in range(size\_x):

matrix[x,0]=x

for y in range(size\_y): matrix[0,y]=y

for x in range(1,size\_x): for y in range(1,size\_y):

if s1[x-1]==s2[y-1]:

matrix[x,y]=min(matrix[x-1,y]+1,matrix[x-1,y-1],matrix[x,y-1]+1) else:

matrix[x,y]=min(matrix[x-1,y]+1,matrix[x-1,y-1]+1,matrix[x,y-1]+1)

print(matrix) return(matrix[size\_x-1,size\_y-1])

s1="Anuja" s2="Anoja"

print levenshelin(s1,s2)

## Output:

runfile('C:/Users/ckt/prac3ir.py', wdir='C:/Users/ckt') [[0. 0. 0. 0. 0. 0.]

[0. 0. 1. 1. 1. 1.]

[0. 1. 0. 1. 2. 2.]

[0. 1. 1. 1. 2. 3.]

[0. 1. 2. 2. 1. 2.]

[0. 1. 2. 3. 2. 1.]]

1.0

# PRACTICAL NO. 4

## Aim: Write a program to Compute Similarity between two text documents.

**Theory**

The cosine similarity is the cosine of the angle between two vectors. Figure 1 shows three 3- dimensional vectors and the angles between each pair. In text analysis, each vector can represent a document. The greater the value of θ, the less the value of cos θ, thus the less the similarity between two documents.

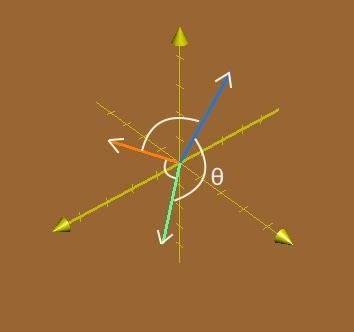
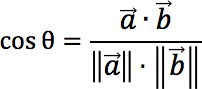


Figure 1.Three 3-dimensional vectors and the angles between each pair. Blue vector: (1, 2, 3); Green vector: (2, 2, 1); Orange vector: (2, 1, 2).

In math equation:



where cosine is the dot/scalar product of two vectors divided by the product of their Euclidean norms.

## Program:

from sklearn.feature\_extraction.text import CountVectorizer from sklearn.metrics.pairwise import cosine\_similarity from sklearn.feature\_extraction.text import TfidfVectorizer f=open('text1.txt','r')

doc1=str.decode(f.read(),"UTF-8","ignore") f=open('text2.txt','r') doc2=str.decode(f.read(),"UTF-8","ignore") train\_set=["president of India",doc1,doc2] tfidf\_vectorizer=TfidfVectorizer()

tfidf\_matrix\_train=tfidf\_vectorizer.fit\_transform(train\_set) #finds the tfidf score with normalization

print "cosine scores ==> ",cosine\_similarity(tfidf\_matrix\_train[0:1], tfidf\_matrix\_train) #here the first element of tfidf\_matrix\_train is matched with other three elements

## Output:

runfile('C:/Users/CKT/prac4 IR.py', wdir='C:/Users/CKT') cosine scores ==> [[1. 1. 1.]]

# PRACTICAL NO. 5

## Aim: Write a map-reduce program to count the number of occurrences of each alphabetic character in the given dataset.

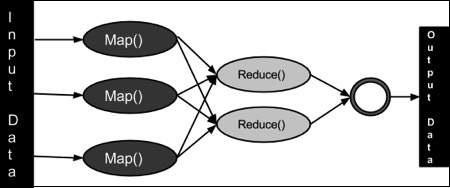
**Theory**

What is MapReduce?

MapReduce is a processing technique and a program model for distributed computing based on java. The MapReduce algorithm contains two important tasks, namely Map and Reduce. Map takes a set of data and converts it into another set of data, where individual elements are broken down into tuples (key/value pairs). Secondly, reduce task, which takes the output from a map as an input and combines those data tuples into a smaller set of tuples. As the sequence of the name MapReduce implies, the reduce task is always performed after the map job.

The Algorithm

* Generally MapReduce paradigm is based on sending the computer to where the data resides!
* MapReduce program executes in three stages, namely map stage, shuffle stage, and reduce stage.
  + Map stage − The map or mapper‟s job is to process the input data. Generally the input data is in the form of file or directory and is stored in the Hadoop file system (HDFS). The input file is passed to the mapper function line by line. The mapper processes the data and creates several small chunks of data.
  + Reduce stage − This stage is the combination of the Shufflestage and the Reduce stage. The Reducer‟s job is to process the data that comes from the mapper. After processing, it produces a new set of output, which will be stored in the HDFS.
* During a MapReduce job, Hadoop sends the Map and Reduce tasks to the appropriate servers in the cluster.
* The framework manages all the details of data-passing such as issuing tasks, verifying task completion, and copying data around the cluster between the nodes.
* Most of the computing takes place on nodes with data on local disks that reduces the network traffic.
* After completion of the given tasks, the cluster collects and reduces the data to form an
* appropriate result, and sends it back to the Hadoop server.



**Program**

Running hadoop 2.2.7 wordcount example under Windows D:\>cd hadoop-2.7.7

d:\hadoop-2.7.>hdfs namenode -format D:\hadoop-2.7.7>start-dfs

D:\hadoop-2.7.7>start- yarn

starting yarn daemons

#Step1 : create a fold input in hadoop file directory d:\hadoop-2.7.7>hdfs dfs -mkdir /input

#Step 2: create file1.txt and copy in input directory

d:\hadoop-2.7.7>hdfs dfs -copyFromLocal d:/hadoop-2.7.7/file1.txt /input d:\hadoop-2.7.7>hdfs dfs -cat /input/file1.txt

D:\hadoop-2.7.7>hdfs dfs -ls /input Found 1 items

-rw-r--r-- 1 USER supergroup 62 2018-12-30 23:25 /input/file1.txt

#step3 : Code mapper.py and reducer.py and check from command prompt d:\hadoop-2.7.7>Python mapper.py <file1.txt >file2.txt

d:\hadoop-2.7.7>Python reducer.py <file2.txt

What is Hadoop Streaming?

Hadoop Streaming is a utility that comes with the Hadoop distribution. It can be used to execute programs for big data analysis. Hadoop streaming can be performed using languages like Python, Java, PHP, Scala, Perl, UNIX, and many more. The utility allows us to create and run Map/Reduce jobs with any executable or script as the mapper and/or the reducer. For example:

$HADOOP\_HOME/bin/hadoop.jar $HADOOP\_HOME/hadoop- streaming.jar \

-input myInputDirs \

-output myOutputDir \

-mapper /bin/cat \

-reducer /bin/wc

Multiple entries can be specified like this:

-files hdfs://host:fs\_port/user/testfile1.txt,hdfs://host:fs\_port/user/testfile2.txt

#step 4 : execute hadoop-streaming

d:\hadoop-2.7.7>hadoop jar d:\hadoop-2.7.7\share\hadoop\tools\lib\hadoop-streaming- 2.7.7.jar -input /input/file1.txt,/input/file2.txt -output /o -file d:/hadoop- 2.7.7/mapper.py -mapper "python mapper.py" -file d:/hadoop-2.7.7/reducer.py -reducer "python reducer.py"

#step5 : list the output file D:\hadoop-2.7.7>hdfs dfs –ls /o Found 2 items

-rw-r--r-- 1 USER supergroup 0 2019-01-01 19:52 /o/\_SUCCESS

-rw-r--r-- 1 USER supergroup 63 2019-01-01 19:52 /o/part-r-00000

D:\hadoop-2.7.7>hdfs dfs -cat /o/part-r- 00000 are 1

hadoop 2 hello

2 how

1

of 1 to

2

welcome 1 world

1 you 1

D:\hadoop-2.7.7>jps 10932 NameNode

8900 DataNode

3688 Jps

6764 NodeManager

9580 ResourceManager

D:\hadoop-2.7.7>python mapper.py <file1.txt

hello

1 how

1 r

1

u 1

learning 1

IR 1

D:\hadoop-2.7.7>python mapper.py <file2.txt This 1

is 1

mapreduce 1 test 1

file 1 welcome 1 to 1

hadoop 1

D:\hadoop-2.7.7>hdfs dfs -copyFromLocal d:/hadoop-2.7.7/file2.txt /input

D:\hadoop-2.7.7>hadoop jar d:\hadoop-2.7.7\share\hadoop\tools\lib\hadoop-streami ng-2.7.7.jar -input /input/file1.txt,/input/file2.txt -output /or -file d:/hado

op-2.7.7/mapper.py -mapper "python mapper.py" -file d:/hadoop-2.7.7/reducer.py

-reducer "python reducer.py"

19/01/27 22:25:28 WARN streaming.StreamJob: -file option is deprecated, please u se generic option -files instead

# PRACTICAL NO. 6

## Aim: Implement a basic IR system using Lucene.

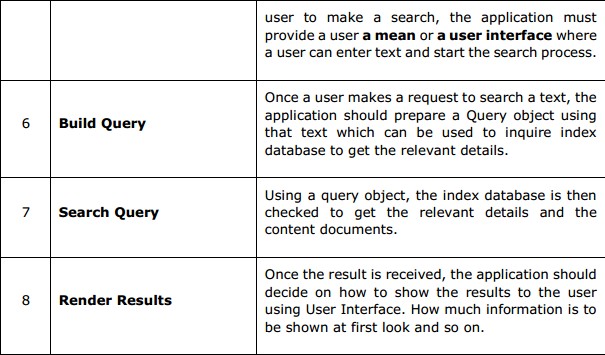
**Theory**

Lucene is a simple yet powerful Java-based Search library. It can be used in any application to add search capability to it. Lucene is an open-source project. It is scalable. This

highperformance library is used to index and search virtually any kind of text. Lucene library provides the core operations which are required by any search application. Indexing and Searching.

A Search a pplication performs a II or a few of the following operations:

|  |  |  |
| --- | --- | --- |
| Step | TiMe | Den<z•iption |
| **1** | **Acquire Raw** Content | The first step of a ny sea rch application is to collect the target contents on which search a pplication is to be conducted . |
| 2 | Build the document | The next step is to bu ild the document(s) from the raw content, which the sea rch application ca n understand and i nterpret easily. |
| **3** | **Analyze the document** | Before the indexi ng process sta rts, the document is to be ana lyzed as to which part of the text is a candidate to be indexed. This process is where the docu ment is ana lyzed. |
| **4** | **indexing the document** | Once documents are built a nd ana lyzed, the next step is to i ndex them so that this document can be retrieved based on certain keys i nstead of the entire content of the document. Indexing process is simila r to indexes in the end of a book where common words are shown with their page numbers so that these words ca n be tracked quickly instead of searching the complete book. |
| **5** | **User Interface for Search** | Once a data base of i ndexes is ready then the application can ma ke a ny search. To facilitate a |



## Steps

Download lucene 7.7.1 Download eclipse Attach 3 jar file

lucene-core-7.7.1.jar

lucene-analyzers-common-7.7.1.jar lucene-queryparser-7.7.1.jar

## Program:

package com.tutorialspoint.lucene; import java.io.IOException;

import org.apache.lucene.analysis.standard.StandardAnalyzer; import org.apache.lucene.document.Document;

import org.apache.lucene.document.Field; import org.apache.lucene.document.TextField; import org.apache.lucene.index.DirectoryReader; import org.apache.lucene.index.IndexReader; import org.apache.lucene.index.IndexWriter;

import org.apache.lucene.index.IndexWriterConfig;

import org.apache.lucene.queryparser.classic.ParseException; import org.apache.lucene.queryparser.classic.QueryParser; import org.apache.lucene.search.IndexSearcher;

import org.apache.lucene.search.Query; import org.apache.lucene.search.TopDocs; import org.apache.lucene.store.Directory;

import org.apache.lucene.store.~~RAMDirectory~~;

public class LuceneHelloWorld {

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

//New index

StandardAnalyzer standardAnalyzer = new StandardAnalyzer(); Directory directory = new ~~RAMDirectory~~();

IndexWriterConfig config = new IndexWriterConfig(standardAnalyzer);

//Create a writer

IndexWriter writer = new IndexWriter(directory, config); Document document = new Document ();

//In a real world example, content would be the actual content that needs to be

indexed.

}

//Setting content to Hello World as an example.

document.add(new TextField("content", "Hello World", Field.Store.*YES*)); writer.addDocument(document);

document.add(new TextField("content", "Hello people", Field.Store.*YES*)); writer.addDocument(document);

writer.close();

//Now let's try to search for Hello

IndexReader reader = DirectoryReader.*open*(directory); IndexSearcher searcher = new IndexSearcher (reader);

QueryParser parser = new QueryParser ("content", standardAnalyzer); Query query = parser.parse("Hello");

TopDocs results = searcher.search(query, 5); System.*out*.println("Hits for Hello -->" + results.totalHits);

//case insensitive search query = parser.parse("hello");

results = searcher.search(query, 5); System.*out*.println("Hits for hello -->" + results.totalHits);

//search for a value not indexed query = parser.parse("Hi there"); results = searcher.search(query, 5);

System.*out*.println("Hits for Hi there -->" + results.totalHits);

}

The above code performs a query on the index using “ Hello ” and “ hello ” as search parameters

* the search returns the total hits as expected.

## Output

Hits for Hello -->2 Hits for hello -->2 Hits for Hi there -->0

**Java code reading file contents and index output on a folder**

package com.tutorialspoint.lucene; import java.io.BufferedReader; import java.io.File;

import java.io.FileReader; import java.io.IOException; import java.nio.file.Paths;

import org.apache.lucene.analysis.standard.StandardAnalyzer; import org.apache.lucene.document.Document;

import org.apache.lucene.document.TextField; import org.apache.lucene.index.DirectoryReader; import org.apache.lucene.index.IndexReader; import org.apache.lucene.index.IndexWriter; import org.apache.lucene.index.IndexWriterConfig;

import org.apache.lucene.index.IndexWriterConfig.OpenMode; import org.apache.lucene.queryparser.classic.ParseException; import org.apache.lucene.queryparser.classic.QueryParser; import org.apache.lucene.search.IndexSearcher;

import org.apache.lucene.search.Query; import org.apache.lucene.search.TopDocs; import org.apache.lucene.store.Directory; import org.apache.lucene.store.FSDirectory; public class LuceneHelloWorldReadFromFile

{

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

{

// New index

StandardAnalyzer standardAnalyzer = new StandardAnalyzer(); String inputFilePath = "C:\\Users\\ckt\\Desktop\\input.txt"; String outputDir = "C:\\Users\\ckt\\Desktop\\output";

File file = new File(inputFilePath);

Directory directory = FSDirectory.*open*(Paths.*get*(outputDir)); IndexWriterConfig config = new IndexWriterConfig(standardAnalyzer); config.setOpenMode(OpenMode.*CREATE*);

// Create a writer

IndexWriter writer = new IndexWriter(directory, config);

Document document = new Document();

try (BufferedReader br = new BufferedReader(new FileReader(inputFilePath)))

{

document.add(new TextField("content", br)); writer.addDocument(document); writer.close();

}

catch (IOException e)

{

e.printStackTrace();

}

// Now let's try to search for Hello

IndexReader reader = DirectoryReader.*open*(directory); IndexSearcher searcher = new IndexSearcher(reader);

QueryParser parser = new QueryParser("content", standardAnalyzer); Query query = parser.parse("Hello");

TopDocs results = searcher.search(query, 5); System.*out*.println("Hits for Hello -->" + results.totalHits);

// case insensitive search query = parser.parse("hello");

results = searcher.search(query, 5); System.*out*.println("Hits for hello -->" + results.totalHits);

// search for a value not indexed query = parser.parse("Hi there"); results = searcher.search(query, 5);

System.*out*.println("Hits for Hi there -->" + results.totalHits);

}

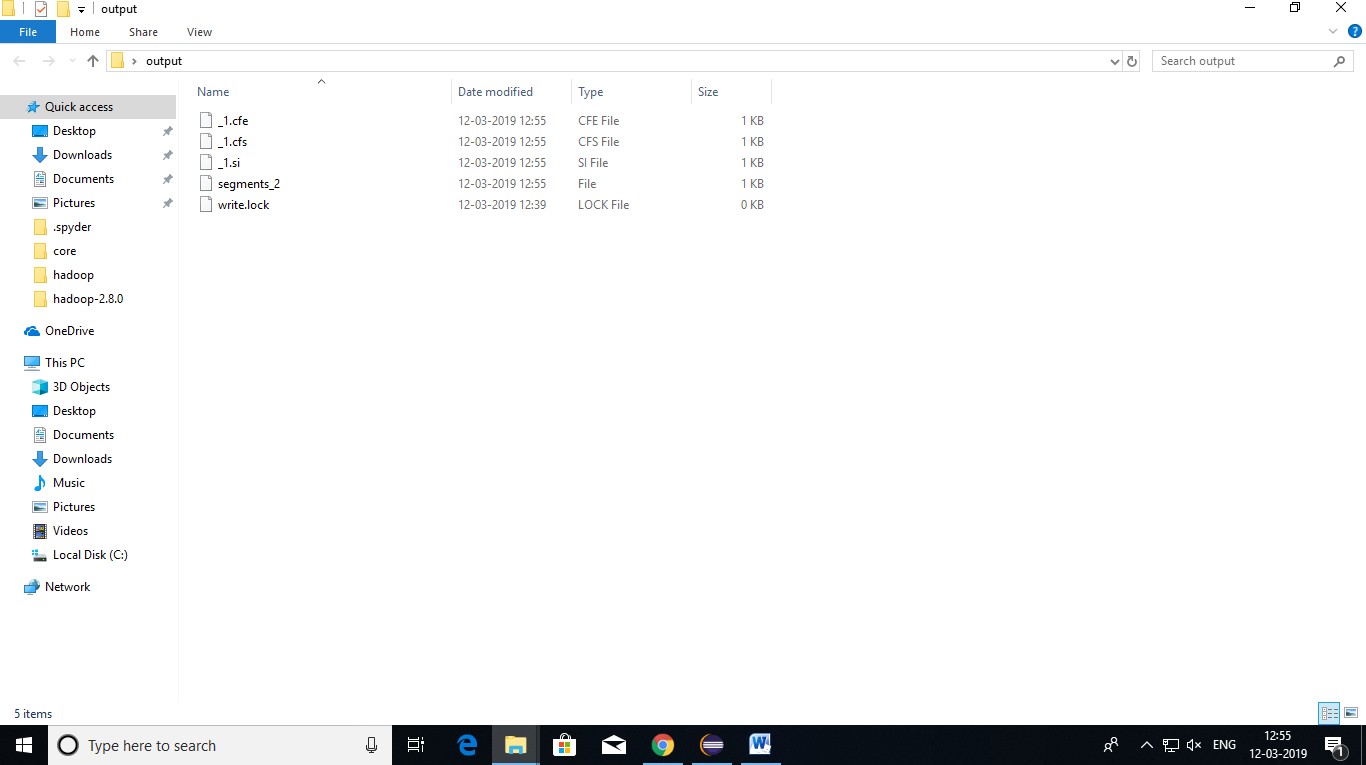
}

## Output

Hits for Hello -->1 Hits for hello -->1 Hits for Hi there -->0

the IndexWriter in this code creates index in the directory presented in attribute outputDir . You

can view the indexing output by viewing the output folder. See a sample output below:



# PRACTICAL NO 7

## Aim: Write program for pre processing of Text document: stop word removal Theory

The process of converting data to something a computer can understand is referred to as pre- processing. One of major forms of pre-processing is to filter out useless data. In natural language processing, useless words(data) are referred to as stop words.

* + What are stop words??

A stop word is commonly used word(such as “the”, “a”, “an”, “in”) that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as a result of search query.

We would not want these words taking up space in our database, or taking up valuable processing time. For this we can remove them easily by sorting a list of words that you consider to be stop words.

NLTK(Natural Language Toolkit) is python has a list of stop words stored in 16 different lamguages. You can find them in nltk-data directory. home/partima/nltk-data/corpora/stopwords is the directory address(Do not forgot to change your home directory name)

To check list stopwords you can type the words following commands in python shell. Import nltk

from nltk.corpus import stopwords set(stopwords.words(„english‟))

{„ourselves‟, „between‟, „yourself‟, „about‟, „out‟, „very‟}

## Program:

from nltk.corpus import stopwords

from nltk.tokenize import word\_tokenize example\_sent= "We are students of CKT college" stop\_words=set(stopwords.words('english')) word\_tokens=word\_tokenize(example\_sent)

filtered\_sentence=[w for w in word\_tokens if not w in stop\_words] filtered\_sentence=[]

for w in word\_tokens:

if w not in stop\_words: filtered\_sentence.append(w)

print(word\_tokens) print(filtered\_sentence)

## Output:

runfile('C:/Users/CKT/prac7.py', wdir='C:/Users/CKT') ['We', 'are', 'students', 'of', 'CKT', 'college']

['We', 'students', 'CKT', 'college']

# PRACTICAL NO. 8

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

**Theory**

Twitter is goldmine of data. Unlike other social platforms, almost every user tweets are completely public and pullable. This is a huge plus if you are trying to get a large amount of data to run analytics on. Twitter data is alsopretty specific. Twitters API allows you to do complex queries like pulling every tweet about a certain topic within the last twenty minutes or pull a certain user is non-retweeted tweets.

* + Twitter Developer Account

In order to use Twitters API we have to create a developer account on Twitter apps site.

1. Login or make a Twitter account at <http://apps.twitter.com/>
2. Create a new app(button on the top right)
3. Fill in the app creation age with a unique name, a website name(use a placeholder website if you don‟t have one) and a project description. Accept the terms and conditions and proceed to next page.
4. Once your project has been created, click on the “keys and access tokens” tab. You should now be able to see your consumer secret and consumer key.
5. You will also need a pair of access tokens, scroll down and request those tokens. The page should refresh and you should now have an access token and access token secret.

## Program:

#Import the necessary methods from tweepy library from tweepy.streaming import StreamListener from tweepy import OAuthHandler

from tweepy import Stream

#Variables that contains the user credentials to access Twitter API consumer\_key = "3yMYKK5Ben0iUaaJ0KGLqrlzk"

consumer\_secret = "gIS4fQrYjpREWxi9RrtgiS4vxzPjlNTIuQmnBCizoL06nrhmNu" access\_token = "1101412887430479872-7YPZMaFXJrR3dRj4BkBHezad9wmJEI" access\_token\_secret = "1a3sXc1OE892iwiEN9GXLLCB2paKkzR1VDBkyvPSbjjDn"

#This is a basic listener that just prints received tweets to stdout. class StdOutListener(StreamListener):

def on\_data(self, data): print data

return True

def on\_error(self, status): print status

if name == ' main ':

#This handles Twitter authetification and the connection to Twitter Streaming API l = StdOutListener()

auth = OAuthHandler(consumer\_key, consumer\_secret) auth.set\_access\_token(access\_token, access\_token\_secret) stream = Stream(auth, l)

#This line filter Twitter Streams to capture data by the keywords: 'python', 'javascript', 'ruby' stream.filter(track=['python', 'javascript', 'ruby'])

## Output:

{"created\_at":"Tue Mar 12 09:23:42 +0000 2019","id":1105398953091035136,"id\_str":"1105398953091035136","text":"RT

@karen73984451: #k9hour #gorgeous girlie ready and waiting on a super duper home #Ruby #Itsallaboutthedogs #TeamZay @epsomcanine plz RT\u2026","source":"\u003ca href=\"http:\/\/twitter.com\/download\/iphone\" rel=\"nofollow\"\u003eTwitter for iPhone\u003c\/a\u003e","truncated":false,"in\_reply\_to\_status\_id":null,"in\_reply\_to\_status\_id\_str

":null,"in\_reply\_to\_user\_id":null,"in\_reply\_to\_user\_id\_str":null,"in\_reply\_to\_screen\_name":null

,"user":{"id":2615014568,"id\_str":"2615014568","name":"Titachot","screen\_name":"Titachot"," location":" Chonburi, Thailand","url":null,"description":"Adopt Don't Shop ! Do not support Puppy Mills

!!","translator\_type":"none","protected":false,"verified":false,"followers\_count":609,"friends\_co unt":654,"listed\_count":5,"favourites\_count":8604,"statuses\_count":120105,"created\_at":"Thu Jul 10 07:38:01 +0000

2014","utc\_offset":null,"time\_zone":null,"geo\_enabled":true,"lang":"th","contributors\_enabled": false,"is\_translator":false,"profile\_background\_color":"F5ABB5","profile\_background\_image\_u rl":"http:\/\/abs.twimg.com\/images\/themes\/theme1\/bg.png","profile\_background\_image\_url\_h ttps":"https:\/\/abs.twimg.com\/images\/themes\/theme1\/bg.png","profile\_background\_tile":true, "profile\_link\_color":"F5ABB5","profile\_sidebar\_border\_color":"C0DEED","profile\_sidebar\_fill

\_color":"DDEEF6","profile\_text\_color":"333333","profile\_use\_background\_image":true,"profil e\_image\_url":"http:\/\/pbs.twimg.com\/profile\_images\/628483133453668352\/KlcJi6Zz\_normal

.jpg","profile\_image\_url\_https":"https:\/\/pbs.twimg.com\/profile\_images\/628483133453668352

\/KlcJi6Zz\_normal.jpg","profile\_banner\_url":"https:\/\/pbs.twimg.com\/profile\_banners\/261501 4568\/1507602801","default\_profile":false,"default\_profile\_image":false,"following":null,"follo w\_request\_sent":null,"notifications":null},"geo":null,"coordinates":null,"place":null,"contributor s":null,"retweeted\_status":{"created\_at":"Mon Mar 11 21:24:18 +0000 2019","id":1105217911927521283,"id\_str":"1105217911927521283","text":"#k9hour #gorgeous

girlie ready and waiting on a super duper home #Ruby #Itsallaboutthedogs #TeamZay @epsomcanine p\u2026 https:\/\/t.co\/90HMoKf7gZ","display\_text\_range":[0,140],"source":"\u003ca href=\"http:\/\/twitter.com\/download\/iphone\" rel=\"nofollow\"\u003eTwitter for iPhone\u003c\/a\u003e","truncated":true,"in\_reply\_to\_status\_id":null,"in\_reply\_to\_status\_id\_str ":null,"in\_reply\_to\_user\_id":null,"in\_reply\_to\_user\_id\_str":null,"in\_reply\_to\_screen\_name":null

,"user":{"id":583791139,"id\_str":"583791139","name":"karen","screen\_name":"karen73984451"

,"location":null,"url":null,"description":"Proud member of #TeamZay, tweet to try to help dogs in rescue find loving forever homes , volunteer with SSPCA

\ud83d\udc36\ud83d\udc36#itsallaboutthedogs#TeamZay","translator\_type":"none","protected": false,"verified":false,"followers\_count":3552,"friends\_count":1711,"listed\_count":334,"favourite s\_count":93937,"statuses\_count":322881,"created\_at":"Fri May 18 13:03:34 +0000 2012","utc\_offset":null,"time\_zone":null,"geo\_enabled":true,"lang":"en","contributors\_enabled": false,"is\_translator":false,"profile\_background\_color":"C0DEED","profile\_background\_image\_u rl":"http:\/\/abs.twimg.com\/images\/themes\/theme1\/bg.png","profile\_background\_image\_url\_h ttps":"https:\/\/abs.twimg.com\/images\/themes\/theme1\/bg.png","profile\_background\_tile":false

,"profile\_link\_color":"1DA1F2","profile\_sidebar\_border\_color":"C0DEED","profile\_sidebar\_fil l\_color":"DDEEF6","profile\_text\_color":"333333","profile\_use\_background\_image":true,"profil e\_image\_url":"http:\/\/pbs.twimg.com\/profile\_images\/1104069307024244738\/iiJJnBgn\_norma l.jpg","profile\_image\_url\_https":"https:\/\/pbs.twimg.com\/profile\_images\/11040693070242447 38\/iiJJnBgn\_normal.jpg","profile\_banner\_url":"https:\/\/pbs.twimg.com\/profile\_banners\/58379 1139\/1538741633","default\_profile":true,"default\_profile\_image":false,"following":null,"follow

\_request\_sent":null,"notifications":null},"geo":null,"coordinates":null,"place":null,"contributors"

:null,"is\_quote\_status":false,"extended\_tweet":{"full\_text":"#k9hour #gorgeous girlie ready and waiting on a super duper home #Ruby #Itsallaboutthedogs #TeamZay @epsomcanine plz RT https:\/\/t.co\/u8Bz8sySMw","display\_text\_range":[0,120],"entities":{"hashtags":[{"text":"k9hou

r","indices":[0,7]},{"text":"gorgeous","indices":[8,17]},{"text":"Ruby","indices":[65,70]},{"text

":"Itsallaboutthedogs","indices":[71,90]},{"text":"TeamZay","indices":[91,99]}],"urls":[],"user\_ mentions":[{"screen\_name":"epsomcanine","name":"Epsom Canine Rescue","id":232647747,"id\_str":"232647747","indices":[100,112]}],"symbols":[],"media":[{"id ":1105217906911178752,"id\_str":"1105217906911178752","indices":[121,144],"media\_url":"ht

tp:\/\/pbs.twimg.com\/media\/D1aF1KTXcAAwZRC.jpg","media\_url\_https":"https:\/\/pbs.twimg

.com\/media\/D1aF1KTXcAAwZRC.jpg","url":"https:\/\/t.co\/u8Bz8sySMw","display\_url":"pic. twitter.com\/u8Bz8sySMw","expanded\_url":"https:\/\/twitter.com\/karen73984451\/status\/11052 17911927521283\/photo\/1","type":"photo","sizes":{"thumb":{"w":150,"h":150,"resize":"crop"}

,"small":{"w":260,"h":240,"resize":"fit"},"large":{"w":260,"h":240,"resize":"fit"},"medium":{"

w":260,"h":240,"resize":"fit"}}}]},"extended\_entities":{"media":[{"id":1105217906911178752, "id\_str":"1105217906911178752","indices":[121,144],"media\_url":"http:\/\/pbs.twimg.com\/med ia\/D1aF1KTXcAAwZRC.jpg","media\_url\_https":"https:\/\/pbs.twimg.com\/media\/D1aF1KTX cAAwZRC.jpg","url":"https:\/\/t.co\/u8Bz8sySMw","display\_url":"pic.twitter.com\/u8Bz8sySM w","expanded\_url":"https:\/\/twitter.com\/karen73984451\/status\/1105217911927521283\/photo

\/1","type":"photo","sizes":{"thumb":{"w":150,"h":150,"resize":"crop"},"small":{"w":260,"h":2

40,"resize":"fit"},"large":{"w":260,"h":240,"resize":"fit"},"medium":{"w":260,"h"

# PRACTICAL NO. 9

## Aim: Write a program to implement simple crawler.

**Theory**

* + Web Crawler

Web crawler is an internet bot that is used for web indexing in World Wide Web. All types of search engines use web crawler to provide efficient results. Actually it collects all or some specific hyperlinks and html content from other websites and preview them in a suitable manner. When there are huge number of links to crawl, even the largest crawler fails. For this reason search engines in early 2000 were bad at providing relevant results, but now this process has improved much and proper results are given in an instance.

## Program:

import re,urllib textfile=file('depth\_1.txt','wt') print"Enter the URL you want to crawl.."

print'Usage - "[http://phocks.org/stumble/creepy/"](http://phocks.org/stumble/creepy/) <--with the double quotes' myurl=input("@>")

for i in re.findall('''href=["'](.[^"']+)["']''',urllib.urlopen(myurl).read(),re.I): print i

for ee in re.findall('''href=["'](.[^"']+)["']''',urllib.urlopen(i).read(),re.I): print ee

textfile.write(ee+'\n')

textfile.close()

## Output:

@>"[http://www.google.com](http://www.google.com/)" <http://www.google.co.in/imghp?hl=en&tab=wi> https:/[/www](http://www.google.co.in/webhp?tab=iw).[google.co.in/webhp?tab=iw](http://www.google.co.in/webhp?tab=iw) <http://maps.google.co.in/maps?hl=en&tab=il> https://play.google.com/?hl=en&tab=i8 <http://www.youtube.com/?gl=IN&tab=i1> <http://news.google.co.in/nwshp?hl=en&tab=in> https://mail.google.com/mail/?tab=im https://drive.google.com/?tab=io https:/[/www](http://www.google.co.in/intl/en/about/products?tab=ih).[google.co.in/intl/en/about/products?tab=ih](http://www.google.co.in/intl/en/about/products?tab=ih) <http://www.google.co.in/history/optout?hl=en>

# PRACTICAL NO. 10

## Aim: Write a program to parse XML text, generate Web graph and compute topic specific page rank.

**Theory**

XML: XML stands for eXtensible Markup Language. It was designed to store and transport data. It was designed to be both human- and machine-readable.That‟s why, the design goals of XML emphasize simplicity, generality, and usability across the Internet.

Parsing XML

We have created parseXML() function to parse XML file. We know that XML is an inherently hierarchical data format, and the most natural way to represent it is with a tree.

Web Graph

The webgraph describes the directed links between pages of the World Wide Web. A graph, in general, consists of several vertices, some pairs connected by edges. In a directed graph, edges are directed lines or arcs.

## Program

import networkx as nx

import matplotlib.pyplot as plt

from xml.dom.minidom import parse import xml.dom.minidom

# Open xml document using minidom parser DOMTree=xml.dom.minidom.parse("movies.xml") collection=DOMTree.documentElement

if collection.hasAttribute("shelf"):

print "Root element: %s" % collection.getAttribute("shelf")

# get all the movies in the collection

movies = collection.getElementsByTagName("movie")

#print detail of each movie. for movie in movies:

print"\*\*\*\*\*Movie\*\*\*\*\*"

if movie.hasAttribute("title"):

print"Title: %s" %movie.getAttribute("title")

type = movie.getElementsByTagName('type')[0] print "Type: %s" % type.childNodes[0].data format= movie.getElementsByTagName('format')[0] print "format: %s" % format.childNodes[0].data rating= movie.getElementsByTagName('rating')[0] print "Rating: %s" % rating.childNodes[0].data

description=movie.getElementsByTagName('description')[0] print"description: %s" % description.childNodes[0].data

def GenerateGraph(): G=nx.Graph()

# adding just one node:

G.add\_node("a")

# adding a list of edges:

G.add\_edges\_from([("a","b"),("b","c"), ("c","d"), ("d","a"),("a","c")])

nx.draw(G)

plt.savefig("simple\_path.png") # save as png plt.show() # display

print("Nodes of graph: ") print(G.nodes()) print("Edges of graph: ") print(G.edges())

GenerateGraph()

## Output

runfile('C:/Users/ckt/.spyder/prct 10 .py', wdir='C:/Users/ckt/.spyder') Root element: New Arrivals

\*\*\*\*\*Movie\*\*\*\*\* Title: Enemy Behind

Type: War, Thriller format: DVD Rating: PG

description: Talk about a US-Japan war

\*\*\*\*\*Movie\*\*\*\*\* Title: Transformers

Type: Anime, Science Fiction format: DVD

Rating: R

description: A schientific fiction

\*\*\*\*\*Movie\*\*\*\*\* Title: Trigun

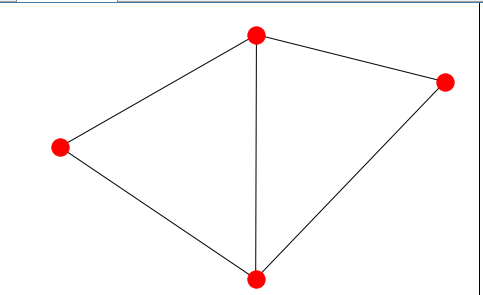
Type: Anime, Action format: DVD Rating: PG

description: Vash the Stampede!

\*\*\*\*\*Movie\*\*\*\*\* Title: Ishtar

Type: Comedy format: VHS Rating: PG

description: Viewable boredom



Nodes of graph:

['a', 'c', 'b', 'd']

Edges of graph:

[('a', 'c'), ('a', 'b'), ('a', 'd'), ('c', 'b'), ('c', 'd')