MAULANA AZAD
NATIONAL INSTITUTE OF TECHNOLOGY
BHOPAL INDIA, 462003
ACADEMIC YEAR 2021-22

WEB SEARCH & WEB MINING
(CSE - 442)

LAB REPORT

BY

Name : JAGDISH RAM TARD

Scholar number : 181112288
CSE-2

TUTORIAL 2

Q. Write a program to implement Term document incidence matrix and inverted index matrix for a given corpus. Program can be written in any programming language (preferably in python, not mandatory). Corpus must be an Input from the user and your program must be dynamic to implement required as per input.

```
#importing libraries
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
import math

#input from user
n=int(input("Enter the number of documents"))
```

```
[11] n=int(input("Enter the number of documents "))

     Enter the number of documents 2
```

```
lis=[]
for i in range(0,n):
  doc=input(" Enter the document ")
  lis.append(doc)
```

```
[12] lis=[]
     for i in range(0,n):
       doc=input(" Enter the document ")
       lis.append(doc)

Enter the document Java is a language for programming that develops a software for several platforms. A compiled code or bytecode on Java application can run on most of the operating
Enter the document Python supports multiple programming paradigms and comes up with a large standard library, paradigms included are object-oriented, imperative, functional and procec
```

```
#DOC 1 : Java is a language for programming that develops a software for several platforms. A compiled code or bytecode on Java application can run on most of the operating systems
```

```
including Linux, Mac operating system, and Linux. Most of the
syntax of Java is derived from the C++ and C languages.

#DOC 2 : Python supports multiple programming paradigms and
comes up with a large standard library, paradigms included are
object-oriented, imperative, functional and procedural.

#Term incidence matrix
my_dict= {}
for i in range(0,len(lis)):
  lis2=[]
  for j in range(0,len(lis[i].split())):
    lis2.append(lis[i].split()[j])
  my_dict[i+1]=([lis[i]])
df=pd.DataFrame(my_dict)
vectorizer = TfidfVectorizer()
doc_vec = vectorizer.fit_transform(df.iloc[0])
df2 =
pd.DataFrame(doc_vec.toarray().transpose(),index=vectorizer.ge
t_feature_names())

for i in range(0,df2.columns.stop):
  lis3=[]
  for j in range(0,len(df2[i])):
    lis3.append(math.ceil(df2[i][j]))

  df2[i]=lis3
for i in range(0,df2.columns.stop):
  df2=df2.rename(columns={i:"Doc"+str(i+1)})
print("Incidence matrix : ")
print(df2)
a_dict={}
```

Incidence matrix :

	Doc1	Doc2
and	1	1
application	1	0
are	0	1
bytecode	1	0
can	1	0
code	1	0
comes	0	1
compiled	1	0
derived	1	0
develops	1	0
for	1	0
from	1	0
functional	0	1
imperative	0	1
included	0	1
including	1	0
is	1	0
java	1	0
language	1	0
languages	1	0
large	0	1
library	0	1
linux	1	0
mac	1	0
most	1	0
multiple	0	1
object	0	1
of	1	0
on	1	0
operating	1	0
or	1	0
oriented	0	1
paradigms	0	1
platforms	1	0
procedural	0	1
programming	1	1
python	0	1
run	1	0
several	1	0
software	1	0
standard	0	1
supports	0	1
syntax	1	0
system	1	0
systems	1	0
that	1	0
the	1	0
up	0	1
with	0	1

```
#Inverted index
for i in range(0,len(lis)):
    for j in range(0,len(lis[i].split())):
      ind=lis[i].split()[j]
      if ind not in a_dict:
        a_dict[ind]=[]
      if ind in a_dict:
        if i+1 not in a_dict[ind]:
          a_dict[ind].append(i+1)
print("Inverted Index : ")
a_dict
```

```
[14] a_dict={}
     for i in range(0,len(lis)):
         for j in range(0,len(lis[i].split())):
           ind=lis[i].split()[j]
           if ind not in a_dict:
             a_dict[ind]=[]
           if ind in a_dict:
             if i+1 not in a_dict[ind]:
               a_dict[ind].append(i+1)
     print("Inverted Index : ")
     a_dict
```

[14]

```
Inverted Index :
{'A': [1],
 'C': [1],
 'C++': [1],
 'Java': [1],
 'Linux,': [1],
 'Linux.': [1],
 'Mac': [1],
 'Most': [1],
 'Python': [2],
 'a': [1, 2],
 'and': [1, 2],
 'application': [1],
 'are': [2],
 'bytecode': [1],
 'can': [1],
 'code': [1],
 'comes': [2],
 'compiled': [1],
 'derived': [1],
 'develops': [1],
 'for': [1],
 'from': [1],
 'functional': [2],
 'imperative,': [2],
 'included': [2],
 'including': [1],
 'is': [1],
 'language': [1],
 'languages.': [1],
 'large': [2],
 'library,': [2],
 'most': [1],
 'multiple': [2],
 'object-oriented,': [2],
 'of': [1],
 'on': [1],
 'operating': [1],
 'or': [1],
 'paradigms': [2],
 'platforms.': [1],
 'procedural.': [2],
 'programming': [1, 2],
 'run': [1],
 'several': [1],
 'software': [1],
 'standard': [2],
 'supports': [2],
 'syntax': [1],
 'system,': [1],
 'systems': [1],
 'that': [1],
 'the': [1],
 'up': [2],
 'with': [2]}
```