INDIAN INSTITUTE OF INFORMATION TECHNOLOGY
LINEAR ALGEBRA -CS305
PROJECT STATUS REPORT

Members:

K. NAVEEN KUMAR(201451074)
SONAL BAIRWA(201451059)
KRITIKA VYAS(201451049)

Topic : Information Retrieval Using Vector spaces.

Work Done on : Modeling Information As a vector and Query Comparison using Geometry , simulation using python.

Approach : In the vector space model, a term by document, or txd, matrix is used to represent the frequency with which certain terms appear in a collection of documents,or a database. Each column of the matrix will represent a different document, and each row will represent a different term. In other words, the documents will make up the column space of the database. The entry aij represents the frequency with which the term of row i appears in the column of document j.

We first ,took a document file and make a txt matrix from it which matches with the entries of the term file. Then we converted it into orthogonal matrix by dividing each column with its norm.

Next coming to Query comparison part we calculated the cosine of the angle between document vector matrix and query vector given by the user.

To test this we will use a query vector searching for "training dogs." The query vector for this will look q = (10100)T .When performing this comparison it is useful to select a cosine value for which any value above is returned as a relevant document. We can refer to this as the cut off value, and we will use a cut off value of .5. Evaluated for each document the cosine values are .5, .8166, .5, and .7071, respectively. Document two is returned as the most relevant of the documents and document four as second most relevant. We also have simulated with some more examples and recorded the output.

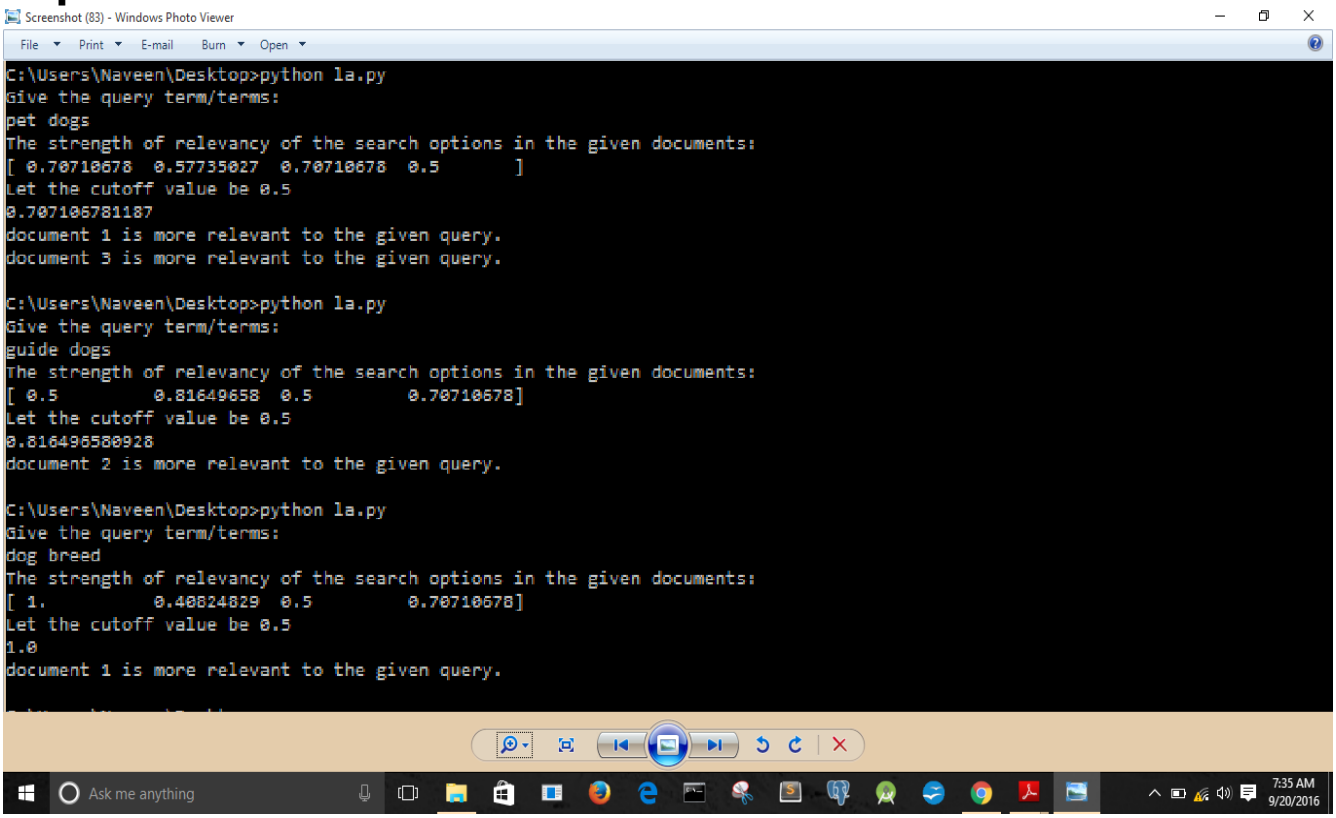
Code:

```
#Modelling information as a vector(VECTOR SPACE MODEL)
import os
import numpy
#given a document file reading it
f1=open('doc.txt','r')
doc=f1.readlines()

#text file
f2=['dog','breed','training','pets','guide']

#extracting each lines of the document into seperate list
doc1=[str(doc[i]).lower() for i in range(len(doc))]

#defining document matrix(initializing them to be zero)
my_matrix=numpy.zeros((len(f2),len(doc1)))

#implementing the matrix which matches with the documents
for j in range(len(doc1)):
        for i in range(len(f2)):
                if(f2[i] in doc1[j] or f2[i][:len(f2[i])-1] in doc1[j]):
                        my_matrix[i][j]=1

#normalizing the matrix(orthogonal matrix)
for i in range(len(doc1)):
        my_matrix[:,i]=my_matrix[:,i]/numpy.linalg.norm(my_matrix[:,i])

#taking query terms from the user
```

```
print 'Give the query term/terms:'
t=str(raw_input().split()).lower()

#query vector
my_array=numpy.zeros((1,len(f2)))
for i in range(len(f2)):
        if(f2[i] in t ):
                my_array[0][i]=1

#print my_array[0]
#finding the angle between the query vector and the document vector,cos,,
(numpy.linalg.norm-->is for finding the norm)
my_final=numpy.dot(my_array,my_matrix)

temp=numpy.linalg.norm(my_array)
temp1=[temp*numpy.linalg.norm(my_matrix[:,i]) for i in range(len(my_final))]

my_final=[my_final[i]/temp1[i] for i in range(len(my_final))]
#print my_final

#printing the final values which shows relevancy,,the cos theta values
print 'The strength of relevancy of the search options in the given
documents:','\n',my_final[0]

#taking cutoff value for reference
print 'Let the cutoff value be 0.5'
k=0.5;

#printing the document with more relevancy
#if max of cos is same as cutoff then printing that no document is more relevant.

flag=0
for i in range(len(doc1)):
        my_final2=max(my_final[0])#finding the max of the cos values the one with
more value is relevant..
        print my_final2
        if my_final2>k:
                break
        else:
                flag=1
```

```
if flag==1:
        print 'none have much relevant information'
else:
        for i in range(len(doc1)):
                if my_final2==my_final[0][i]:
                        print 'document',i+1,'is more relevant to the given query.';
```

output:



This is just one way for query retrieval .Here the system is not always perfect, but there are ways to amend the process and obtain even more accurate results.

Further we would like to reduce the computation using ,SVD ,QR decomposition.

END