**ASSIGNMENT NO.**

NAME: SHRUTI DILIP BHUJANGE

CLASS: BE COMP-1 ROLL NO.: 402006

PROGRAM:

import re, math

def termFrequency(document): # output the normalized term frequencies

text = open(document).read()

reducedDocument = [i for i in re.split('[ .\n]', text.lower()) if i]

termFrequencyDict = {}

for i in reducedDocument:

if i not in termFrequencyDict:

termFrequencyDict[i] = reducedDocument.count(i) / float(len(reducedDocument))

return termFrequencyDict

def inverseDocumentFrequency(allDocuments):

idfSet = set()

idfDict, tempDict = {}, {}

wordSets = []

text = ""

for doc in allDocuments:

text = open(doc).read() # merge all the text data

wordSets.append(set([i for i in re.split('[ .\n]', text.lower()) if i])) # wordSets is a doc.wise freq. count set

idfSet |= wordSets[-1]

for i in idfSet: # initialization of the dictionary

tempDict[i] = 0.0

for i in tempDict:

for j in range(len(wordSets)):

if i in wordSets[j]: # number of documents in which the word has appeared

tempDict[i] += 1

for i in tempDict: # if numDocumentsWithThisTerm > 0:

if tempDict[i] > 0:

idfDict[i] = 1.0 + math.log(float(len(allDocuments))/tempDict[i])

else:

idfDict[i] = 1.0

return idfDict

def tfIdf(tf\_list, idf):

for doc in tf\_list:

for term in tf\_list[doc]:

tf\_list[doc][term] = (tf\_list[doc][term], tf\_list[doc][term]\*idf[term])

def cos\_sim(query, document\_list, idf, tf\_list):

query\_set = set(query.split()) # to eliminate duplication

term\_freq = 1.0/len(query\_set) # the frequency of each term of the query

cosines\_dict = {}

max\_similarity = 0 # initialization of similarity

max\_doc = None

for doc in document\_list:

cos\_sum = 0

query\_mag = 0 # query magnitude and doc. term magnitude

doc\_mag = 0

for term in query\_set:

cos\_sum += (idf[term] if term in idf else 0)\*term\_freq\*(tf\_list[doc][term][1] if term in tf\_list[doc] else 0)

query\_mag += math.pow((idf[term] if term in idf else 0)\*term\_freq, 2)

doc\_mag += math.pow((tf\_list[doc][term][1] if term in tf\_list[doc] else 0), 2)

query\_mag = math.sqrt(query\_mag)

doc\_mag = math.sqrt(doc\_mag)

#print query\_mag, doc\_mag, cos\_sum

#print query\_mag\*doc\_mag+0.001

cos\_sum /= (query\_mag\*doc\_mag+0.001) # to avoid 0/0

if (cos\_sum >= max\_similarity):

max\_similarity = cos\_sum

max\_doc = doc

if (max\_similarity == 0):

max\_doc = "None"

return (max\_similarity, max\_doc)

def main():

document\_list = ["doc-1", "doc-2", "doc-3"]

termFrequencyList = {} # contains the term frequency of all documents's terms

for doc in document\_list:

termFrequencyList[doc] = termFrequency(doc)

idf = inverseDocumentFrequency(document\_list) # inverse document frequency for all terms of all documents

print idf, "\n"

tfIdf(termFrequencyList, idf)

print termFrequencyList

query = raw\_input("Enter the query : ")

max\_sim, doc\_with\_max\_sim = cos\_sim(query, document\_list, idf, termFrequencyList)

print "The max sim. is", max\_sim, "for the doc.", doc\_with\_max\_sim

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

main()

Input:

doc-1: The game of life is a game of everlasting learning.

doc-2: The unexamined life is not worth living.

doc-3: Never stop learning.