import nltk

from nltk.corpus import stopwords

import math

from nltk.stem import PorterStemmer

from nltk.corpus import PlaintextCorpusReader

#finding stopwords of english from stopwords corpus

stop=stopwords.words('english')

#Loading own corpus in nltk. corpus\_project containing documents to be processed

newcorpus = PlaintextCorpusReader(r"\Users\Parul\Desktop\corpus\_project", ".\*")

#fileids of corpus

print(newcorpus.fileids())

#words in all documents

for f in newcorpus.fileids():

print(newcorpus.words(f))

#stop word removal

#stemming words of document using Porter Stemmer

ps=PorterStemmer()

word=[ps.stem(w) for w in newcorpus.words() if w not in stop and w.isalpha()]

#taking query to be searched as input

query=input("enter the query")

#Writing input query in file

query\_file=open(r"C:\Users\Parul\Desktop\QueryFolder\queryfile.txt","w")

query\_file.write(query)

query\_file.close()

#Loading corpus of file containing input query

querycorpus = PlaintextCorpusReader(r"\Users\Parul\Desktop\QueryFolder", ".\*")

#finding words of query

#removing stopwords from query

#stemming query words using porter stemmer

wordQuery=[ps.stem(w) for w in querycorpus.words() if w not in stop and w.isalpha()]

#Conditional frequency distribution of query file after stopword removal and stemming

cfdist = nltk.ConditionalFreqDist(

(fileid, word)

for fileid in querycorpus.fileids()

for word in wordQuery)

cfdist.tabulate()

#Conditional frequency distribution of documents with all words after stopword removal and stemming

#vector space model using term frequency

cfd = nltk.ConditionalFreqDist(

(fileid, w)

for fileid in newcorpus.fileids()

for w in [ps.stem(w) for w in newcorpus.words(fileid) if w not in stop and

w.isalpha()])

cfd.tabulate()

countquery=0;

for w in set(word):

countquery=countquery+(cfdist['queryfile.txt'][w]\*cfdist['queryfile.txt'][w])

countquery=math.sqrt(countquery)

mul=[]

for f in newcorpus.fileids():

count=0;

counter=0;

for w in set(word):

#print(word)

count=count+(cfd[f][w]\*cfdist['queryfile.txt'][w])

counter=counter+(cfd[f][w]\*cfd[f][w])

mul.append(count/(math.sqrt(counter) \* countquery))

#Compute similiarity of each document with query

#store values of similiarity in dictionary with key as document name and value as similiarity value

result={}

for f in newcorpus.fileids():

count=0;

counter=0;

for w in set(word):

#print(word)

count=count+(cfd[f][w]\*cfdist['queryfile.txt'][w])

counter=counter+(cfd[f][w]\*cfd[f][w])

result[f]=count/(math.sqrt(counter) \* countquery)

#sort result as deascending order of similiarity(values)

import operator

sorted\_d={}

sorted\_d=sorted(result.items(),key=operator.itemgetter(1),reverse=True)