# IR programming assignment #2

The report to TA
B06406009 資管三 陳姵如

**環境：python3、安裝下載nltk**

## 1. 先import library

```
import re
import os
import io
import math
import string
import numpy
from nltk import PorterStemmer
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
```

## 2.讀檔

```
#read the files
file = []
text = []
a = []
b = 0
num_of_file = 1095

for i in range (0, num_of_file):
    b = b+1
    a.append(str(b) + '.txt')

for i in range (0, num_of_file):
    file.append(open(a[i], 'r'))
    text.append(file[i].read())
    file[i].close()
```

## 3.資料前處理

```
#lowcast
# Tokenization
# Poter's Algorithm
# Stopword removal
# Redundancy removal
```

```
for i in range(0, num_of_file):
    text[i] = text[i].lower()

stop_words = set(stopwords.words('english'))
example_sent = []
word_tokens = []

for i in range(0, num_of_file):
    word_tokens.append(word_tokenize(text[i]))


filtered_sentence = []
text_term = [] #是一個雙層list，分文本存裡面挑過的單詞

for i in range(0, num_of_file):
    for w in word_tokens[i]:
        if w not in stop_words and w.isalpha()  and filter(lambda 
x: x.isalpha(), w) != "":
            filtered_sentence.append(PorterStemmer().stem(w))
    text_term.append(filtered_sentence)
    filtered_sentence = []
```

**4.建立、輸出字典、計算df**

```
# Establish my dictionary
# Sort my dictionary

my_dictionary = []

for i in range (0, num_of_file):
    for j in range (len (text_term[i]) ):
        if text_term[i][j] not in my_dictionary:
            my_dictionary.append (text_term[i][j])

my_dictionary.sort()

# Calculate the df
# Output dictionary

df = []

for i in range (0, len (my_dictionary)):
    df.append(0)


for i in range (0, len(my_dictionary)):
    for j in range(0, num_of_file):
        if my_dictionary[i] in text_term[j]:
            df[i] += 1
```

```
f = open('dictionary.txt','w')
for i in range (0, len(my_dictionary)):
    f.write(str(i)+ " " + my_dictionary[i]+ " " + str(df[i]) +
'\n')
f.close()
```

## 5.計算tf、idf、tf-idf

```
# Calculate the tf
# Calculate the idf
# Calculate the tf-idf

tf = [] #每個詞在“每個文檔”出現幾次
idf = []
tf_idf = []
text_tmp = ""
tf_tmp = []
tf_idf_tmp = []

for i in range(0, num_of_file):
    for j in range (0, len(text_term[i])):
        tf_tmp.append(0)
        text_tmp = text_term[i][j]
        for k in range (0, len(text_term[i])):
            if (text_tmp == text_term[i][k]):
                tf_tmp[j] += 1
    tf.append(tf_tmp)
    tf_tmp = []


for i in range (0, len (my_dictionary)):
    idf.append(0)

for i in range (0, len (my_dictionary)):
    idf[i] = math.log(num_of_file / df[i] , 10)


for j in range(0, num_of_file):
    for k in range (0, len(text_term[j])):
        for i in range (0, len(my_dictionary)):
            if text_term[j][k] == my_dictionary[i]:
                tf_idf_tmp.append(tf[j][k] * idf[i])
    tf_idf.append(tf_idf_tmp)
    tf_idf_tmp = []
```

## 6.輸出vector files、檔名是Doc1.txt/Doc2.txt...

```
# Save the unit vector files

c = 0
d = []

for i in range (0, num_of_file):
    c = c+1
    d.append( 'Doc' + str(c) + '.txt')
# print (d[i])
# print( len(my_dictionary) )

for i in range (0, num_of_file):
    f = open( d[i] ,'w')
    f.write ( str(len(text_term[i])) +'\n' )
    for j in range (0, len(text_term[i]) ):
        for k in range (0, len(my_dictionary)):
            if text_term[i][j] == my_dictionary[k]:
                f.write( str(k) + " " + str(tf_idf[i][j]) + '\n' )
    f.close()
```

## 7.寫了一個cosine relation table 可以檢查後面的 cosine function

```
# A cosine relation table

vector = [] #一個檔案對應一個向量（維度＝字典詞數）#雙層迴圈
vector_tmp = []
cosine = []
cosine_tmp = []
multi_sum = 0
lenx = 0
leny = 0

for j in range(0, num_of_file):
    for i in range (0, len(my_dictionary)):
        vector_tmp.append(0)
        for k in range ( 0, len(text_term[j]) ):
            if my_dictionary[i] == text_term[j][k]:
                vector_tmp[i] = tf_idf[j][k]
    vector.append(vector_tmp)
    vector_tmp = []
# print(vector)

# print( len(vector), len(vector[0]) )
for i in range(0, num_of_file):
    for j in range(0, num_of_file):
        cosine_tmp.append(0)
        for k in range (0, len(my_dictionary)):
            multi_sum += vector[i][k]*vector[j][k]
            lenx += vector[i][k]*vector[i][k]
            leny += vector[j][k]*vector[j][k]
        cosine_tmp[j] = multi_sum / 
( math.sqrt(lenx)*math.sqrt(leny) )
        multi_sum = 0
        lenx = 0
        leny = 0
    cosine.append( cosine_tmp )
    cosine_tmp = []
# print(cosine)

#Write a function cosine(Docx, Docy) which loads the tf-idf 
vectors of documents x and y and returns their cosine similarity.
```

## 8.傳入兩個vector就會return兩者的cosine similarity

```
def cosine(docx, docy): #預設維度一樣

    multi_sum = 0
    lenx = 0
    leny = 0
    cos = 0

    for i in range(0, len(docx)):
        multi_sum += docx[i]*docy[i]
        lenx += docx[i]*docx[i]
        leny += docy[i]*docy[i]
    cos = multi_sum / (math.sqrt(lenx) * math.sqrt(leny))

    return cos
```