

# HW2 Report

R07725021 資管碩 1 洪靖雯

## 1.執行環境

- Sublime
- Cmder

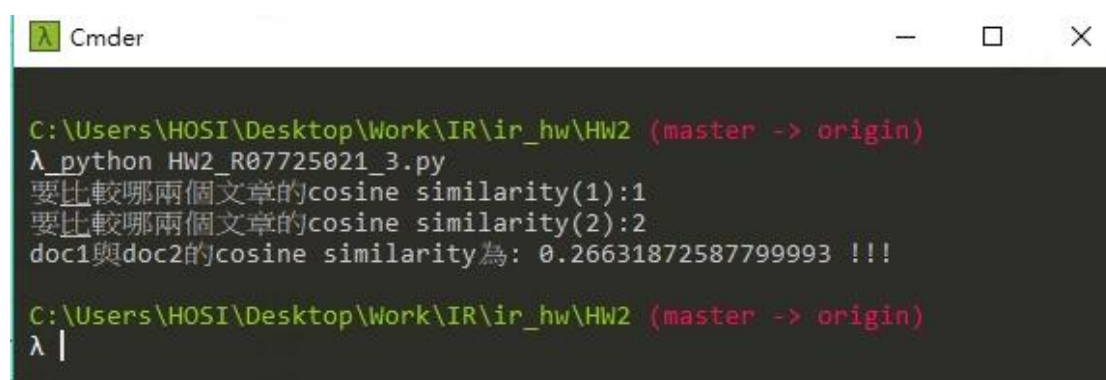
## 2.程式語言

- Python 3.6.6

## 3.執行方式

- 套件: requests、bs4、nltk
  - 有另外用 conda 4.5.8 架虛擬環境安裝，直接用在 system 裡應該也可行
  - 使用 pip install 方式安裝
    - ◆ pip install requests
    - ◆ pip install bs4
    - ◆ pip install nltk
- 編譯方式: 這次共有三個檔案，HW2\_R07725021\_1.py 是做將 documents tokenize 後計算每個 term 的 df 並儲存到 dictionary.txt

裡、HW2\_R07725021\_2.py 是算出每個 term 的 tf-idf unit vector 並各別輸出到對應的檔案 x.txt 裡面、HW2\_R07725021\_3.py 是根據前面 tf-idf 結果讓使用者輸入要比較的兩個 document，算出這兩個 document 的 cosine similarity 並輸出到畫面上。所以，如只是要算出兩個檔案 cosine similarity，只需在 cmd(cmd)中輸入 python HW2\_R07725021\_3.py 即可執行。



```
C:\Users\HOSI\Desktop\Work\IR\ir_hw\HW2 (master -> origin)
λ python HW2_R07725021_3.py
要比較哪兩個文章的cosine similarity(1):1
要比較哪兩個文章的cosine similarity(2):2
doc1與doc2的cosine similarity為: 0.26631872587799993 !!!

C:\Users\HOSI\Desktop\Work\IR\ir_hw\HW2 (master -> origin)
λ |
```

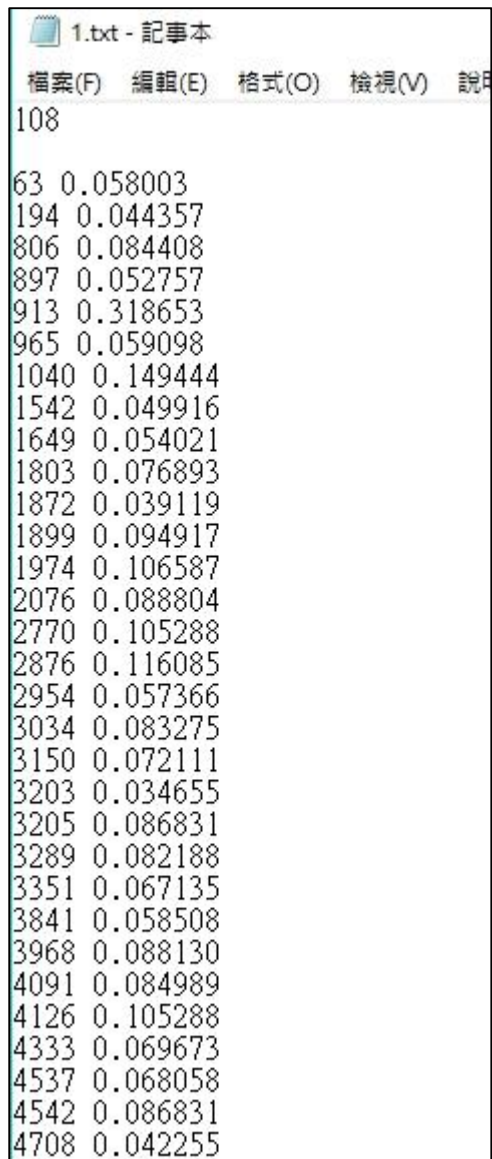
## ➤ 輸出結果

### i. dictionary.txt (總共 11678 個 term)



```
dictionary.txt - 記事本
檔案(F) 編輯(E) 格式(O) 檢視(V) 說明(H)
1 aan 1
2 aaron 2
3 aback 1
4 abahd 1
5 abandon 37
6 abat 1
7 abc 49
8 abdallah 2
9 abdel 3
10 abdomen 2
11 abduct 2
12 abdul 5
13 abdullah 40
14 abdurahman 1
15 aberr 1
16 abhad 1
17 abhiyan 1
18 abhorr 2
19 abid 7
20 abidjan 36
21 abijan 1
```

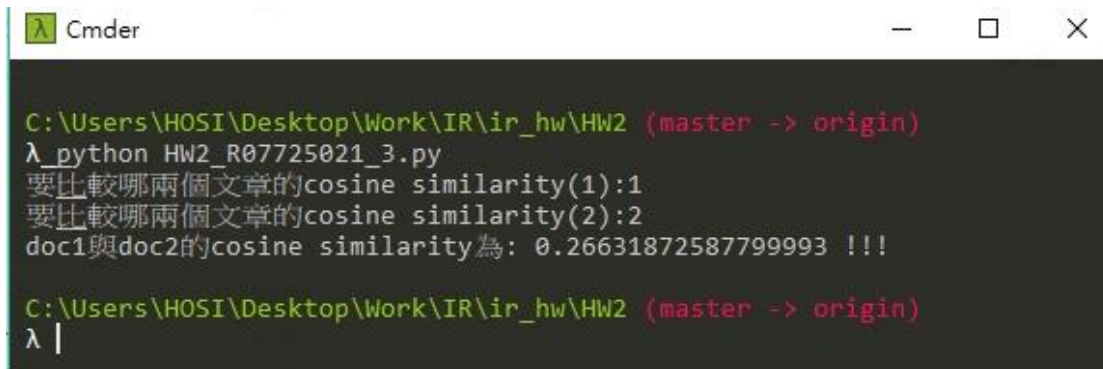
ii. 1.txt (the vector file of document)



108

63	0.058003
194	0.044357
806	0.084408
897	0.052757
913	0.318653
965	0.059098
1040	0.149444
1542	0.049916
1649	0.054021
1803	0.076893
1872	0.039119
1899	0.094917
1974	0.106587
2076	0.088804
2770	0.105288
2876	0.116085
2954	0.057366
3034	0.083275
3150	0.072111
3203	0.034655
3205	0.086831
3289	0.082188
3351	0.067135
3841	0.058508
3968	0.088130
4091	0.084989
4126	0.105288
4333	0.069673
4537	0.068058
4542	0.086831
4708	0.042255

iii. cosine similarity 輸出在 cmd 上



```
C:\Users\HOSI\Desktop\Work\IR\ir_hw\HW2 (master -> origin)
λ python HW2_R07725021_3.py
要比較哪兩個文章的cosine similarity(1):1
要比較哪兩個文章的cosine similarity(2):2
doc1與doc2的cosine similarity為: 0.26631872587799993 !!!

C:\Users\HOSI\Desktop\Work\IR\ir_hw\HW2 (master -> origin)
λ |
```

#### 4.作業處理邏輯說明

因為有三個 script，所以以下分別介紹

##### i. HW2\_R07725021\_1.py

運用 HW1 的 tokenize 方法，將 document 切割取得數個 term，計算各個 term 的 df 值後輸出到 dictionary.txt 中。

➤ Import 必要的 package

```
import requests
from bs4 import BeautifulSoup
import re
from nltk.tokenize import word_tokenize
from nltk.stem import PorterStemmer
#import nltk
#nltk.download('punkt') #do this once
```

➤ 蒐集 1095 個 document 內容

```
9 #initialize
10 dict_all = {}
11
12 # 讀取document collection
13 for i in range(1,1096):
14     file = open("IRTM/"+str(i)+".txt","r")
15     # file to string array
16     data = file.read()
17     file.close()
18
```

➤ 引用 HW1 script 來做 tokenize，並改良了 stopword removal 方法

(因為原先方法是去網路抓，這樣等網路回應太慢，所以改由抓下來儲存為 stopwordlist.txt，讀取檔案來判斷)

```

9  #initialize
10 dict_all = {}
11 dict_tf = {}
12
13 # 讀取document collection
14 for i in range(1,1096):
15     file = open("IRTM/"+str(i)+".txt","r")
16     # file to string array
17     data = file.read()
18     file.close()
19
20     #tokenize(改用nltk的tokenizer)
21     tokendata = word_tokenize(data)
22     regex = [c for c in tokendata if c.isalpha()] #去除non-alphanumeric
23
24     #lowercasing
25     lowerString = [elements.lower() for elements in regex]
26
27     #stemming
28     stemmer = PorterStemmer()
29     stemString = [stemmer.stem(lowerstring) for lowerstring in lowerString]
30
31     #stopword removal
32     #因為會有回應時間過長問題，直接抓下來讀取檔案
33     file2 = open("stopwordlist.txt","r")
34     stopwordList = file2.read()
35     file2.close()
36     removeStopword = [word for word in stemString if word not in stopwordList]
37

```

- 用一個 dictionary 型態 `dict_now` 儲存目前這個 document 的 term 和其 tf(key 為 term，value 為 tf)，並將結果輸出在 `TermFrequency/x.txt` 中(為了第二個程式用的)。而後也用一個 dictionary 型態的 `dict_all` 來儲存全部 documents 的 term 和 df(key 為 term,value 為 df)，用來與 `dict_now` 比較目前有沒有儲存 `dict_now` 的 term，如果有就在 `dict_all` 的 value 加一，沒有就儲存該 term。因此就能獲得 1095 documents 所有的 term 和總共出現次數 df!

```

37     #儲存到dictionary
38     dict_now = {} #儲存該doc的dict
39     for word in removeStopword:
40         if(dict_now.get(word)): #有key, value+1
41             dict_now[word] += 1
42         else: #否則在dictionary加上該key, 並從value=1開始
43             dict_now[word] = 1
44
45     #將每一個doc的term的tf先輸出成txt(ex. 1.txt...)
46     with open('TermFrequency/'+str(i)+'.txt','w+') as f: #w: write
47         for key,value in sorted(dict_now.items()):
48             f.write("%s %d\n" % (key,value))
49
50     #用dict_now儲存全部, dict_now儲存每一個doc的
51     #判斷dict_now是否已經存有該term
52     for key,value in dict_now.items():
53         if(dict_all.get(key)):
54             dict_all[key] += 1
55         else:
56             dict_all[key] = 1
57

```

- 最終將 dict\_now 排序過，並加上 t\_index(按照 sort 過的 dictionary 排順序，從 1 開始)以及 term 和 df 輸出到 dictionary.txt 檔案上，格式為 t\_index、term、df。

```

58 # 將terms次數印到txt檔上, 格式為t_index、term、df, 並且是排
59 with open('dictionary.txt','w') as f:
60     t_index = 1
61     for key,value in sorted(dict_all.items()):
62         f.write("%d %s %d\n" % (t_index,key,value))
63         t_index = t_index+1
64

```

## ii. HW2\_R07725021\_2.py

這步驟主要做的是計算每個 term 的 tf-idf 值，並做 normalization(除以 vector length)，變成 tf-idf unit vector.

- 讀取 dictionary.txt 裡面儲存的 term 和 df 值，用 dict\_df 儲存；另讀取 TermFrequency/ 各檔案裡面的 term 和 tf 值，用 dict\_tf 儲存。



```

18 dict_df = {} #儲存dictionary的term和df
19 dict_index = {} #儲存dictionary的term和t_index
20 #取得dictionary.txt的值轉成dict
21 with open('dictionary.txt','r') as f:
22     for line in f:
23         (t_index,key,value) = line.split()
24         dict_df[key] = value #儲存term,df
25         dict_index[key] = t_index #儲存term,t_index
26
27 #讀取TermFrequency每一個doc的term的tf
28 for i in range(1,1096):
29     dict_now = {} # 儲存該doc的term和tf
30
31     # doc i's file to dictionary
32     with open('TermFrequency/'+str(i)+'.txt','r') as f:
33         for line in f: #一行一行儲存
34             (key,value) = line.split()
35             dict_now[key] = value
36

```

- 將準備輸出 term 和 tf-idf 結果的檔案 TF\_IDF 做清理(因為後面輸出方式是用 append，所以需要在這裡清理)。先輸出該 document 有多少的 term 在第一行。

```

37 # 先清除i.doc檔案，寫上該doc含有多少term
38 open('TF_IDF/'+str(i)+'.txt','w').close() # clear doc contents
39 with open('TF_IDF/'+str(i)+'.txt','w') as f:
40     f.write(str(len(dict_now))+"\n\n") # how many terms in this doc
41

```

- 定義一個 function `countDocVectorLength`，輸入一個 dictionary，計算該 dictionary 的 vector length.

$$(|V(d1)| = \sqrt{d1[0]^2 + d1[1]^2 + d1[2]^2 + \dots d1[n]^2})$$

```

10 def countDocVectorLength(dict): #計算vector length
11     squareNum = 0
12     for key,value in dict.items():
13         squareNum += int(value) * int(value) #tf^2相加
14     sqrtNum = math.sqrt(squareNum) #最後squart roots
15     return sqrtNum
16
41
42 vectorLength = countDocVectorLength(dict_now) #計算vector length

```

- 另外，`import math`，定義一個 function `findIDF`，輸入 `dict_df`、`term`、`tf`，取得 `dict_df` 的 `df` 值換成 `idf` 值( $idf = \log_{10}(N/df)$ )，與輸入的 `tf` 值後相乘得 `tf-idf` 值( $tf\_idf = tf * idf$ )。

得到 `tf-idf` 值後，做 `normalize`，將 `tf-idf` 值除以前面得到的 `vector length`。最終得到 `tf-idf unit vector`。

```
1 import math
2
3 def findIDF(dict,key,value): #去dictionary.txt
4     #在dictionary.txt找到該term的df，將df變idf
5     df = dict.get(key)
6     idf = math.log10(1095/float(df))
7     tf_idf = float(value)*float(idf)
8     return float(tf_idf)
9
43 for key,tf in dict_now.items(): #count tf-idf
44     tf_idf = findIDF(dict_df,key,tf)
45     t_index = int(dict_index.get(key))
46     sqrt_tfidf = float(tf_idf/vectorLength) #
47
```

- 輸出結果到 `TF_IDF/x.txt`(各 document 檔案)，輸出 `t_index`、`tf-idf unit vector`

```
with open('TF_IDF/'+str(i)+'.txt','a') as f: #
    f.write("%d %f\n" % (t_index,sqrt_tfidf))
```

### iii. HW2\_R07725021\_3.py

算出每個 document `tf-idf` 值後，給 user 輸入要比較兩個 document，得到這兩個 document 的 `cosine similarity`，最後輸出在 `cmd` 上。

- 詢問要比較的 2 個 document ID，定義一個 function `readFileToDict`，



輸入 document 的代號，從 TF\_IDF/x.txt 中取得該 document 的 term、tf-idf 值，並轉成 dictionary 格式回傳。

```
22 doc1 = input("要比較哪兩個文章的cosine similarity(1):")
23 doc2 = input("要比較哪兩個文章的cosine similarity(2):")
24 dict_1 = readFileToDict(doc1)
25 dict_2 = readFileToDict(doc2)

1 def readFileToDict(num): #選擇要load哪個doc，並將內容換
2     with open('TF_IDF/'+str(num)+'.txt','r') as f:
3         dict = {}
4         for index,line in enumerate(f,start=1): #(x,y
5             if(index>=3):
6                 #print("line:%s" %line)
7                 (key,value) = line.split()
8                 dict[int(key)] = float(value)
9         return dict
10
```

- 另外定義一個 function cosineSimilarity，輸入兩個 document 的 dictionary，比較兩個 document 是否含有相同的 term，有則相乘 tf-idf 的值，最後把所有各 term 相乘的 tf-idf 值相加，得到這兩個 document 的 cosine similarity 值，輸出到 cmd 上。

```
11 def cosineSimilarity(dict1,dict2):
12     count = 0
13     for key_1 in dict1: #比較dict2是否有包含dict
14         if key_1 in dict2:
15             value_1 = dict1[key_1] #dict1的valu
16             value_2 = dict2[key_1] #dict2的valu
17             count += float(value_1*value_2) #兩
18     return count #回傳結果
19
```

```
26 answer = cosineSimilarity(dict_1,dict_2)
27 print("doc"+str(doc1)+"與doc"+str(doc2)+"的cosine similarity為: "+str(answer)+" !!!")
```