IR HW2 資管碩二 R05725034 張鑑霖

1. 執行環境 & 作業系統

   Jupyter & win7

2. 程式語言

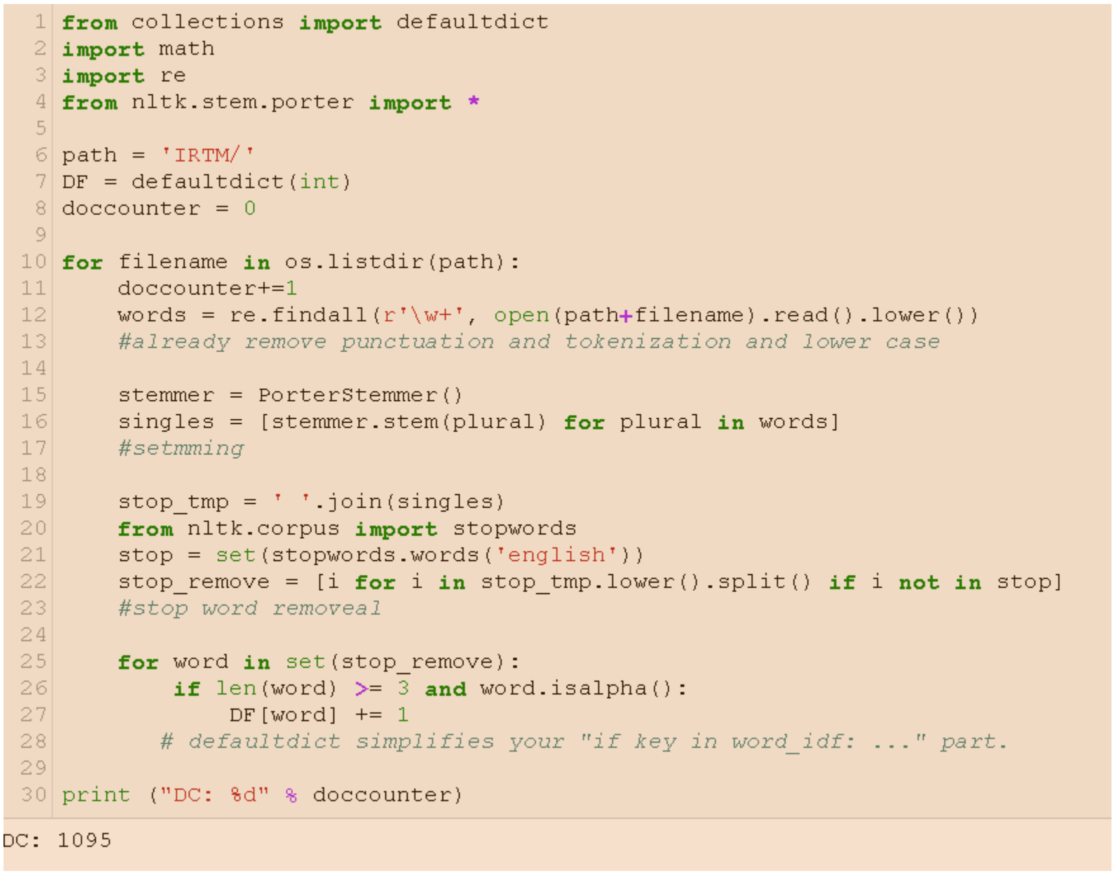
   Python3

3. 作業處理邏輯說明

   甲、 Construct a dictionary:

   i. 讀入每個在資料夾 IRTM 的檔案

      並去除標點、tokenization、小寫化、stemming、去除 stop word，並記錄有幾筆文件(1095 筆)，如果字長度大於 3 和是英文，就把此文字的計數器+1，因為用的是 set，所以處理時每份文件不會計算重複的字。

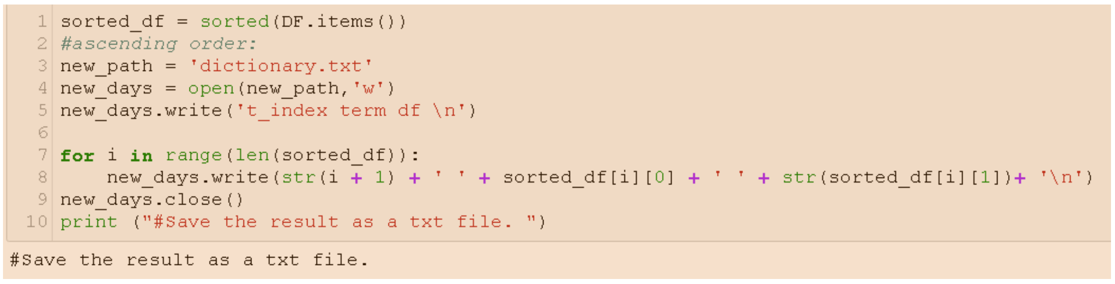


```
from collections import defaultdict
import math
import re
from nltk.stem.porter import *

path = 'IRTM/'
DF = defaultdict(int)
doccounter = 0

for filename in os.listdir(path):
    doccounter+=1
    words = re.findall(r'\w+', open(path+filename).read().lower())
    #already remove punctuation and tokenization and lower case

    stemmer = PorterStemmer()
    singles = [stemmer.stem(plural) for plural in words]
    #setmming

    stop_tmp = ' '.join(singles)
    from nltk.corpus import stopwords
    stop = set(stopwords.words('english'))
    stop_remove = [i for i in stop_tmp.lower().split() if i not in stop]
    #stop word removeal

    for word in set(stop_remove):
        if len(word) >= 3 and word.isalpha():
            DF[word] += 1
        # defaultdict simplifies your "if key in word_idf: ..." part.

print ("DC: %d" % doccounter)
```

```
DC: 1095
```

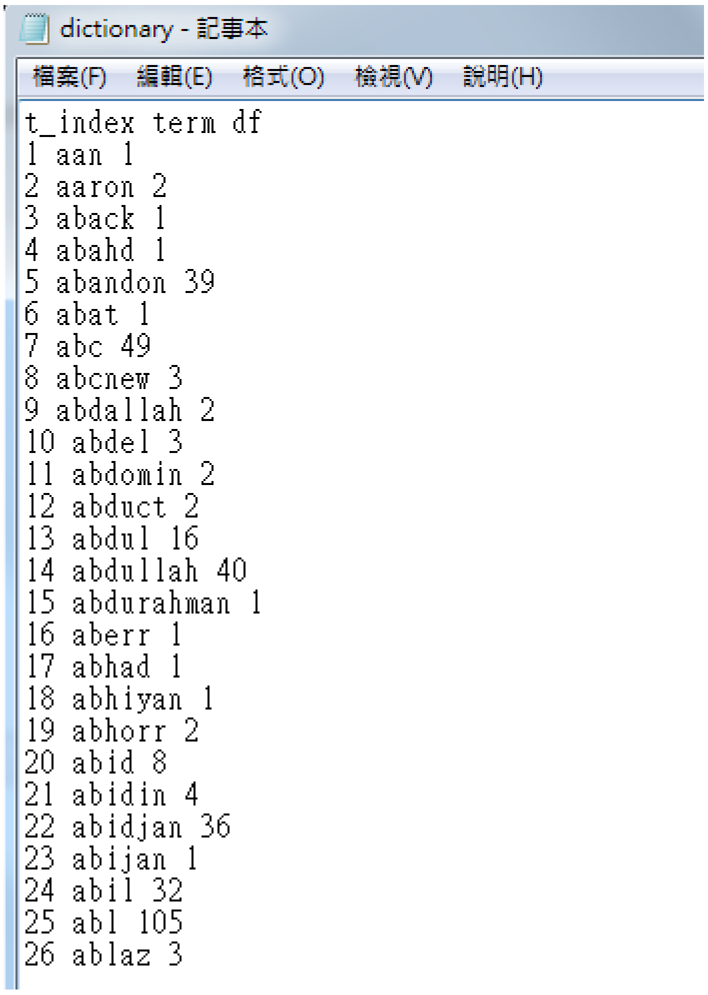
   ii. 輸出

      把紀錄的文字 - 計數依照文字排序，並存入 dictionary.txt



```
sorted_df = sorted(DF.items())
#ascending order:
new_path = 'dictionary.txt'
new_days = open(new_path,'w')
new_days.write('t_index term df \n')

for i in range(len(sorted_df)):
    new_days.write(str(i + 1) + ' ' + sorted_df[i][0] + ' ' + str(sorted_df[i][1])+ '\n')
new_days.close()
print ("#Save the result as a txt file. ")
```

```
#Save the result as a txt file.
```



```
t_index term df
1 aan 1
2 aaron 2
3 aback 1
4 abahd 1
5 abandon 39
6 abat 1
7 abc 49
8 abcnew 3
9 abdallah 2
10 abdel 3
11 abdomin 2
12 abduct 2
13 abdul 16
14 abdullah 40
15 abdurahman 1
16 aberr 1
17 abhad 1
18 abhiyan 1
19 abhorr 2
20 abid 8
21 abidin 4
22 abidjan 36
23 abijan 1
24 abil 32
25 abl 105
26 ablaz 3
```

乙、 Tfidf unit vector

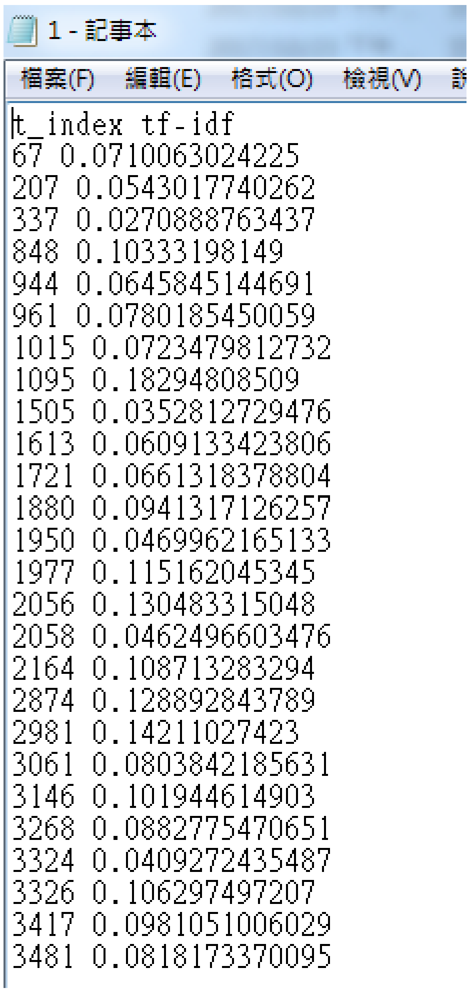
i. 計算每個文字的 idf(log 以 10 為底)，並排序

```python
# Now you can compute IDF.
IDF = dict()
for word in DF:
    IDF[word] = math.log10(doccounter / float(DF[word]))

sorted_idf = sorted(IDF.items())
#print (sorted_tfidf)

```

ii. 開起每個在資料夾 IRTM 的檔案，做一樣的前處理

得到每個 TF，每個 TF 再乘上剛剛相對應的 IDF，再標準化即可得到結果，檔案存於 tfidf 資料夾內。

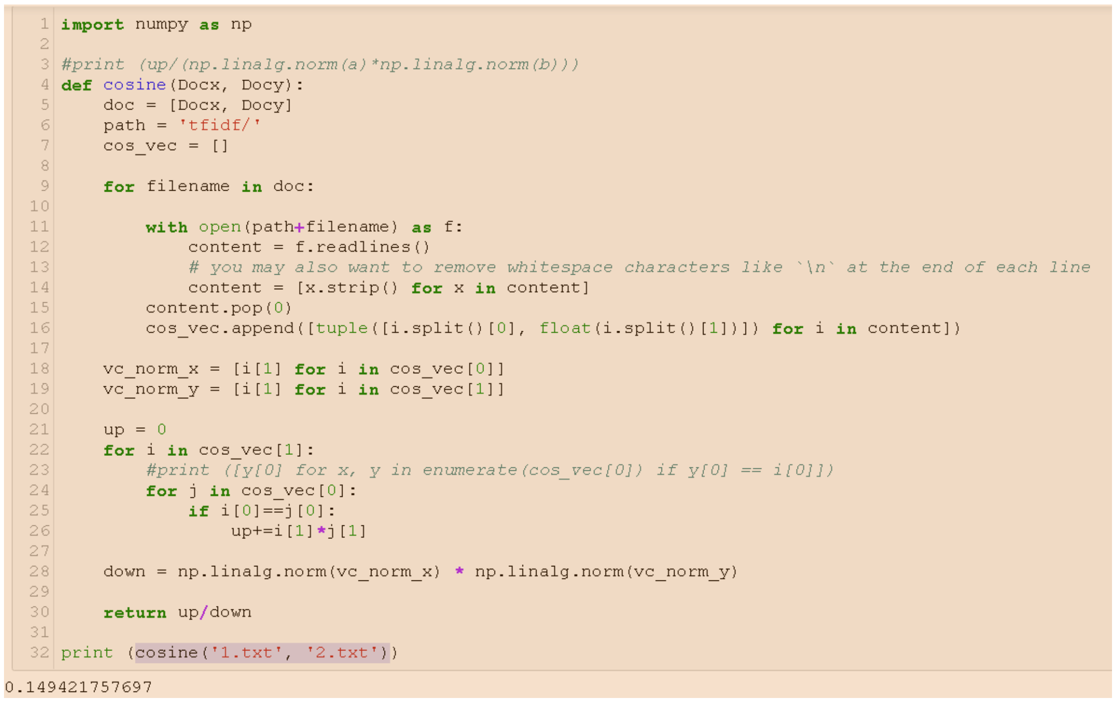
```python
import numpy as np
#doccounter = 1
for filename in os.listdir(path):
    tmp = defaultdict(float)
    new_path = 'tfidf/'+filename
    new_days = open(new_path,'w')
    new_days.write('t_index tf-idf \n')
    #doccounter+=1

    words = re.findall(r'\w+', open(path+filename).read().lower())
    #already remove punctuation and tokenization and lower case

    stemmer = PorterStemmer()
    singles = [stemmer.stem(plural) for plural in words]
    #setmming

    stop_tmp = ' '.join(singles)
    from nltk.corpus import stopwords
    stop = set(stopwords.words('english'))
    stop_remove = [i for i in stop_tmp.lower().split() if i not in stop]
    #stop word removeal

    for word in set(stop_remove):
        if len(word) >= 3 and word.isalpha():
            tmp[word] += 1


    sp = sorted(tmp.items())
    unorm=[]
    for i in range(len(sp)):
        #y= idf, sp = 1.txt

        unorm.append(sp[i][1] * IDF[sp[i][0]])

    for i in range(len(sp)):
        ide = [x+1 for x, y in enumerate(sorted_idf) if y[0] == sp[i][0]]
        new_days.write(str(ide[0]))

        new_days.write(' ' + str(sp[i][1] * IDF[sp[i][0]] / np.linalg.norm(unorm))+'\n')
    new_days.close()
    print (filename)
```



```
t_index tf-idf
67 0.0710063024225
207 0.0543017740262
337 0.0270888763437
848 0.10333198149
944 0.0645845144691
961 0.0780185450059
1015 0.0723479812732
1095 0.18294808509
1505 0.0352812729476
1613 0.0609133423806
1721 0.0661318378804
1880 0.0941317126257
1950 0.0469962165133
1977 0.115162045345
2056 0.130483315048
2058 0.0462496603476
2164 0.108713283294
2874 0.128892843789
2981 0.14211027423
3061 0.0803842185631
3146 0.101944614903
3268 0.0882775470651
3324 0.0409272435487
3326 0.106297497207
3417 0.0981051006029
3481 0.0818173370095
```

丙、 function *cosine*

i. 輸入兩個文件名計算 cosine similarity，先讀入問題”乙”的檔案，並計算兩個檔案的 norm 當分母，而分子只需要算兩份文件都有出現的 term，並把他們的 tfidf 相乘後相加即可，例如 cosine('1.txt', '2.txt') =0.149421757697



```python
import numpy as np

#print (up/(np.linalg.norm(a)*np.linalg.norm(b)))
def cosine(Docx, Docy):
    doc = [Docx, Docy]
    path = 'tfidf/'
    cos_vec = []

    for filename in doc:

        with open(path+filename) as f:
            content = f.readlines()
            # you may also want to remove whitespace characters like `\n` at the end of each line
            content = [x.strip() for x in content]
        content.pop(0)
        cos_vec.append([tuple([i.split()[0], float(i.split()[1])]) for i in content])

    vc_norm_x = [i[1] for i in cos_vec[0]]
    vc_norm_y = [i[1] for i in cos_vec[1]]

    up = 0
    for i in cos_vec[1]:
        #print ([y[0] for x, y in enumerate(cos_vec[0]) if y[0] == i[0]])
        for j in cos_vec[0]:
            if i[0]==j[0]:
                up+=i[1]*j[1]

    down = np.linalg.norm(vc_norm_x) * np.linalg.norm(vc_norm_y)

    return up/down

print (cosine('1.txt', '2.txt'))
```

```
0.149421757697
```

4. 任何在此作業中的心得

   有些地方會執行緩慢，可能有地方可以優化執行效率，是我可以努力的目標，而且在處理這三小題中，有些地方感覺用到的變數也是有重複，如果能更佳善用可以節省記憶體空間，也可以運算得更快速