

## 作業二

### 1. 執行環境

本機開 Anaconda，建立虛擬環境，下 python 去跑.py 檔。

用 Visual Studio Code 撰寫 python 程式碼。

### 2. 程式語言 (請標明版本)

Python 3.7.11

### 3. 執行方式

本機開啟 Anaconda Powershell Prompt，Anaconda 版本是 conda 4.10.3，



Anaconda Powershell Prompt (anaconda)

應用程式

使用 nltk 與 numpy 套件。

- 創建虛擬環境：(以下我有先創虛擬環境，但也可以不創建)

```
$conda create --name IR python=3.7
```

下 **\$conda env list** 可以看到剛剛創建的虛擬環境。

```
(base) PS C:\Users\...> conda env list
# conda environments:
#
base                * D:\anaconda
AIHW1               D:\anaconda\envs\AIHW1
IR                  D:\anaconda\envs\IR
pyhon3.7            D:\anaconda\envs\pyhon3.7
```

用 **\$ conda activate IR** 可以啟動剛剛創好的虛擬環境。

- 安裝 nltk 套件：(若沒有 nltk 套件一定需要 install)

然後由於有用到 nltk 套件抓 stopwords list，

所以要下 **\$conda install -c anaconda nltk**。

- 安裝 numpy 套件：(若沒有 numpy 套件一定需要 install)

有用到 numpy 的功能所以要 install。

下 `$pip install numpy`。

```
(IR) PS D:\jupyter\IR_HW\pa2> pip install numpy
Collecting numpy
  Downloading numpy-1.21.4-cp37-cp37m-win_amd64.whl (14.0 MB)
    | 14.0 MB 1.1 MB/s
Installing collected packages: numpy
Successfully installed numpy-1.21.4
(IR) PS D:\jupyter\IR_HW\pa2>
```

然後可以透過 `$conda list` 看目前環境中載的所有套件。

其他套件是在 create 虛擬環境時有問要不要安裝，我當時選 yes。

```
(IR) PS D:\jupyter\IR_HW\pa2> conda list
# packages in environment at D:\anaconda\envs\IR:
#
# Name                    Version            Build    Channel
ca-certificates           2020.10.14         0        anaconda
certifi                    2020.6.20          py37_0   anaconda
click                      7.1.2              py_0     anaconda
joblib                     0.17.0             py_0     anaconda
nltk                       3.5                py_0     anaconda
numpy                      1.21.4             pypi_0   pypi
openssl                   1.1.1i             h2bbff1b_0
pip                        21.2.4             py37haa95532_0
python                    3.7.11             h6244533_0
regex                      2020.10.15         py37he774522_0
setuptools                58.0.4             py37haa95532_0
sqlite                    3.36.0             h2bbff1b_0
tqdm                      4.50.2             py_0     anaconda
vc                         14.2               h21ffa451_1
vs2015_runtime            14.27.29016        h5e58377_2
wheel                     0.37.0             pyhd3eb1b0_1
wincertstore              0.2                py37haa95532_2
```

- 執行程式：

然後去到放 pa2.py 的資料夾，`$python .\pa2.py` 就可以執程式碼，結果會如下圖，最後會印出 doc1 跟 doc2 的 cosine similarity。

```
[nltk_data] Package stopwords is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]   C:\Users\Annie\AppData\Roaming\nltk_data...
[nltk_data] Package stopwords is already up-to-date!
Finish generatedic
Cosine similarity of doc1 and doc2: 0.2864153874748629
```

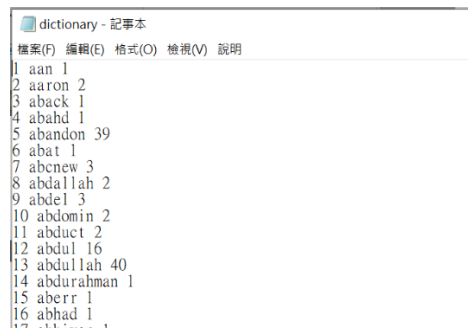
並在相同資料夾下產生一個 dictionary.txt 檔。

```
(IR) PS C:\Users\Annie\Downloads\R10725011> ls

目錄: C:\Users\Annie\Downloads\R10725011

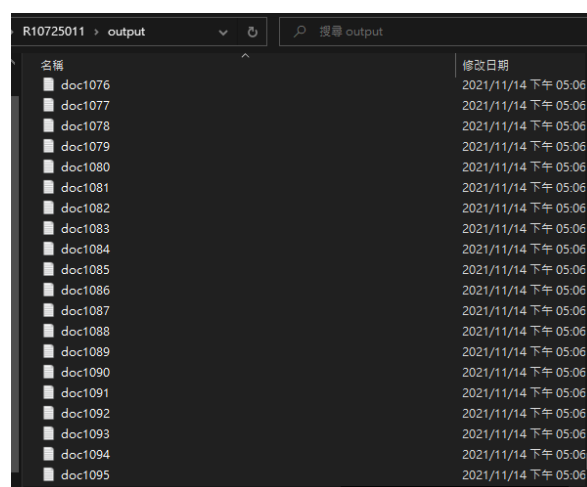
Mode                LastWriteTime         Length Name
----                -
d-----          2021/11/14 下午 04:38             data
d-----          2021/11/14 下午 05:06             output
-a-----          2021/11/14 下午 05:06          192344 dictionary.txt
-a-----          2021/11/14 下午 05:06           4160 pa2.py
```

dictionary.txt 點開如下圖，是 t\_index term df 的形式。



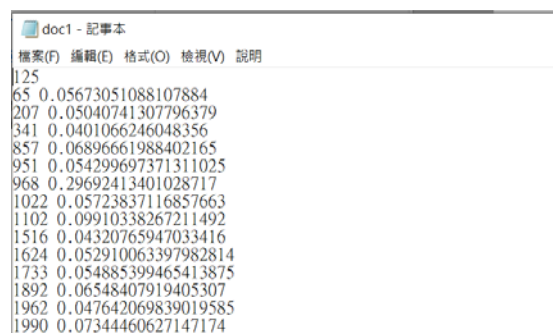
```
dictionary - 記事本
檔案(F) 編輯(E) 格式(O) 檢視(V) 說明
1: aan 1
2: aaron 2
3: aback 1
4: abahd 1
5: abandon 39
6: abat 1
7: abcnew 3
8: abdallah 2
9: abdel 3
10: abdomin 2
11: abduct 2
12: Abdul 16
13: abdullah 40
14: abdurahman 1
15: aberr 1
16: abhad 1
```

同時會在 output 資料夾中產生 1095 個(根據 data 有多少 document)doc\*.txt 檔，如下圖：



點開 doc1.txt 為例：

第一行顯示共有 125 個 term，之後是 t\_index tf-idf 的形式。



```
doc1 - 記事本
檔案(F) 編輯(E) 格式(O) 檢視(V) 說明
125
65 0.05673051088107884
207 0.05040741307796379
341 0.0401066246048356
857 0.06896661988402165
951 0.054299697371311025
968 0.29692413401028717
1022 0.05723837116857663
1102 0.09910338267211492
1516 0.04320765947033416
1624 0.052910063397982814
1733 0.054885399465413875
1892 0.06548407919405307
1962 0.047642069839019585
1990 0.07344460627147174
```

#### 4. 作業處理邏輯說明

- import

包含用來抓取資料的 glob、os，用來抓 stopwords list 的 nltk，用來做 Porter's algorithm 的 PorterStemmer，用來作矩陣運算的 numpy 跟數學運算

的 math。

```
import nltk
import glob
import os
import math
import numpy as np
from nltk.stem import PorterStemmer
```

- 大綱

```
# Generate Dictionary
```

主程式一開始先呼叫 generatedic function，裡面會呼叫 tokenize function 做上次作業做的 Lowercasing、Tokenization、Stemming 與 Remove Stopwords 的處理，其中會用 TFtds 紀錄每個 doc 的 tf，也會產出有 df 的 dictionary，最後產生 dictionary.txt 並回傳 dictionary list d 與 TFtds 給主程式。

```
# Transfer each document into a tf-idf unit vector
```

利用前面回傳的 d 與 TFtds 算出 tf\_idfs，並求出每個 doc 的 unit vector，產生 1095 個 doc\*.txt 於 output 資料夾中。

```
# Cosine similarity.
```

呼叫 cosine(Docx, Docy) 可以得到 Docx 與 Docy 的 cosine similarity。Docx 與 Docy 為 int 代表的是 Doc index，像是要求 doc1 與 doc2 的 cosine similarity 就可以呼叫 cs = cosine(1, 2)，cs 即為 cosine similarity 的值。

- 詳細說明

```
# Generate Dictionary
if __name__ == '__main__':
    # Generate Dictionary
    d, TFtds = generatedic()
    print("Finish generatedic\n")
```

進入主程式呼叫 generatedic()

```
def generatedic():
    text = "" #用來抓資料
    dic = {} #用來存 term 跟對應的 df
    TFtds = {} #用來存所有 doc 的 term 與對應的 tf，如下圖：
```

```

1.txt: {'white': 1, 'hous': 1, 'also': 1, 'keep': 1, 'close': 1, 'watch': 1, 'yugoslavia': 1, 'opposit': 5, 'forc': 2,
step': 1, 'pressur': 2, 'presid': 1, 'slobodan': 2, 'milosev': 6, 'work': 4, 'nbc': 2, 'jim': 2, 'maceda': 2, 'belgra
d': 5, 'tonight': 1, 'serbia': 2, 'eve': 1, 'gener': 2, 'strike': 4, 'two': 1, 'hour': 1, 'roadblock': 1, 'tast': 1, 'co
me': 2, 'tomorrow': 2, 'say': 2, 'nationwid': 1, 'stoppag': 1, 'begin': 1, 'purpos': 1, 'man': 1, 'stole': 1, 'president
i': 1, 'elect': 1, 'peopl': 2, 'accept': 1, 'victim': 1, 'hostag': 1, 'one': 1, 'person': 1, 'time': 3, 'four': 1, 'year
': 1, 'ago': 1, 'hundr': 1, 'thousand': 1, 'serb': 1, 'march': 1, 'call': 1, 'well': 1, 'protest': 1, 'last': 1, 'three'
: 2, 'month': 1, 'surviv': 1, 'real': 1, 'question': 1, 'power': 1, 'confront': 1, 'regim': 2, 'combin': 1, 'wealth': 1,
'get': 1, 'rid': 1, 'case': 1, 'point': 1, 'coal': 1, 'mine': 1, 'heart': 1, 'economy': 1, 'hand': 1, 'worker': 1, 'gua
rd': 1, 'equip': 1, 'week': 1, 'electr': 1, 'reserv': 1, 'engin': 1, 'worri': 1, 'blackout': 1, 'make': 2, 'gestur': 1,
'tri': 1, 'sens': 1, 'difficulti': 1, 'spread': 1, 'word': 1, 'independ': 1, 'radio': 2, 'station': 1, 'shut': 1, 'signa
l': 1, 'jam': 1, 'leav': 1, 'journalist': 1, 'frustrat': 1, 'wait': 1, 'better': 1, 'hear': 1, 'listen': 1, 'poorli': 1,
'organ': 1, 'ralli': 1, 'shrink': 1, 'draw': 1, 'bare': 1, 'leader': 1, 'hope': 1, 'chang': 1, 'civil': 1, 'disobed':
1, 'throughout': 1, 'land': 1, 'may': 1, 'give': 1, 'new': 1, 'life': 1, 'struggl': 1, 'oust': 1, 'news': 1})
2.txt: {

```

```

def key_func(x): #用來按 data 檔案的數字順序抓取資料
    txt = os.path.split(x)[1]
    num = int(txt.split('.')[0])
    return num

# Read text, Tokenize, Get tf and df of the doc
#用 sorted by key=key_func 來按 data 檔案的數字順序抓取資料
for filename in sorted(glob.glob('.\\data\\*.txt'),key=key_func ):
    with open(os.path.join(os.getcwd(), filename), 'r') as f:
        text = f.read()

        # Tokenize doc
#這邊是上次作業的範圍對 doc 進行 tokenize、stemming、remove stopwords 等等
#因為不是此次作業主要內容，將詳細說明放到最後面的補充
        result = tokenize(text)
#記錄這個 doc 的所有 term 跟 term 在此 doc 發生的次數 tf 到 TFtds 中
        # Get tf of the doc
        TFtds[os.path.split(filename)[1]] = {i:result.count(i) for i in
result}

#算出 term 發生在幾個 doc 中：df，記錄在 dic 裡
#沒出現過就新增一個，有出現過就+1 次
#因為用 set 處理過所以不會重複在同一個 doc 算到兩次一樣的 term

        # Get df of the doc
        result = set(result) #用 set 來將 result 裡面元素唯一
        result = list(result)
        for t in result:
            if dic.get(t, 0) == 0:
                dic[t] = 1
            else:
                dic[t] += 1

#利用 dic 中的第 0 個元素就是 term 來做排序(按 alphabet 排序 dictionary)
# Sorted dictionary by terms
dics = sorted(dic.items(), key=lambda x:x[0])

```

```

# Generate dictionary.txt and return dictionary list d, TFtds back
dictionary = "" #用來產生 dictionary.txt
d = [] #dictionary 的 list 形式，會回傳
cnt = 0 #用來設置 dictionary term index
#產生 dictionary.txt
for i in dics:
    cnt += 1
    c = str(cnt)
    i1 = str(i[1])
    dictionary += c + " " + i[0] + " " + i1 + "\n"
    d.append([c,i[0],i[1]])
# Output dictionary txt
with open('.\\dictionary.txt','w') as f:
    f.write(dictionary)
#回傳 dictionary list d 與 TFtds 回去
return d, TFtds

```

接下來回到主程式。

利用剛剛得到的 d 與 TFtds 來產生每個 doc 的 tf-idf unit vector。

```

# Transfer each document into a tf-idf unit vector
#算出每個 doc 的 tf_idf unit vector
for i in range(len(TFtds)):
    doc_unit = [] #用來放 tf-idf unit vector
    doc_index = [] #用來放 term index
    doc = "" #用來產生 doc*.txt
    cnt = 0 #用來記錄這個 doc 有多少 term
    #抓出某份 doc 的所有 term 跟 tf
    filename = str(i+1)+'.txt'
    TFtd = TFtds[filename]
    #用 for 迴圈跑 dictionary 為了抓到 term index 跟算 tf_idf
    for j in range(len(d)):
        t_index = d[j][0]
        term = d[j][1]
        df = d[j][2]
        N = len(d)
        #如果跑到的 dictionary term 在這份 doc 中，才去算 tf_idf
        if term in TFtd:

```

```

    cnt += 1
    tf = TFtd[term]
    idf = math.log(N/df, 10) #根據課堂投影片公式算出 idf
    tf_idf = tf * idf #算出 tf_idf
    doc_index.append(t_index) #將 term index 記錄下來
    doc_unit.append(tf_idf) #將 tf_idf 記錄下來
#輸出的成 doc*.txt 的第一行是總共有幾個 term，為 doc 加上第一行
    doc = str(cnt) + '\n' + doc
#利用 np.linalg.norm 算出 vector 長度  $|V(d_i)|$ 
#根據  $\frac{V(d_i)}{|V(d_i)|}$  算出這個 doc 的 unit vector
    doc_unit = doc_unit / np.linalg.norm(doc_unit)
#產生 doc*.txt
    for k in range(len(doc_unit)):
        doc += str(doc_index[k]) + ' ' + str(doc_unit[k]) + '\n'
    with open('.\\output\\doc'+ filename, 'w') as f:
        f.write(doc)

```

接下來是最後一步算出兩個 doc vector 的 cosine similarity。

```

# Cosine similarity.
cs = cosine(1, 2)
print("Cosine similarity of doc1 and doc2: ", cs)

```

在主程式呼叫 cosine(Docx, Docy)，會回傳答案回來。

這邊按作業要求實際執行 doc1 與 doc2 的 cosine similarity。

```

def cosine(Docx, Docy):
    x = [] #用來存 docx 的 unit vector 值
    y = [] #用來存 docy 的 unit vector 值
    dlen = 0 #用來記錄 dictionary 有幾個 term
#讀取前面建好的 dictionary.txt 並記錄共有幾個 term
    with open('.\\dictionary.txt', 'r') as f:
        for line in f:
            dlen += 1
#用來讀前面產生的 doc*.txt，處理後回傳化成 list 形式的 doc
    def readfile(Doc):
        Dlist = []
        with open('.\\output\\doc' + str(Doc) + '.txt', 'r') as f:
            for line in f:

```







補充：

基本上跟第一次作業一樣，多的地方是多去掉數字跟單一字母，並將 stopwords list stem 後再次進行去除。

```
def tokenize(text):
    # Remove punctuations in text
    punc = '!'()-[]{};"'\,<>?@#$$%^&*~`''
    urlpunc = '"/.:''
    num = '1234567890''
    #去掉標點符號、數字換成空格
    for e in text:
        if e in punc or e in urlpunc or e in num:
            text = text.replace(e, ' ')
    #去掉回車換行
    # Remove \r\n
    text = text.replace('\r','').replace('\n','')
    #大寫變小寫
    # Lowercasing
    text = text.lower()
    #按空格拆分
    # Tokenization
    sptext = text.split(' ')
    #用 porterStemmer 做 stemming
    # Stemming using Porter's algorithm
    ps = PorterStemmer()
    smtext = []
    for t in sptext:
        smtext.append(ps.stem(t))
    #從 nltk 下載 stopwords list
    # Remove Stopwords
    # Using stopwords list from nltk
    nltk.download('stopwords')
    stop_words = nltk.corpus.stopwords.words('english')
    # print(stop_words)
    #對 stopwords 做 stemming
    smstop = []
    for s in stop_words:
        for i in s:
```

```
        if i in punc:
            s = s.replace(i, '')
        smstop.append(ps.stem(s))

result = []
sw = []
alpha = ''abcdefghijklmnopqrstuvwxyz''
#去除 stopwords 跟 stemming 後的 stopwords 跟單個字母跟空格
for t in smtext:
    if not t in stop_words and not t in smstop and not t in alpha and
t != '':
        result.append(t)
    else:
        sw.append(t)
#回傳整理好的 result
return result
```