資訊檢索與文字探勘導論
Homework 2

資管三吳驊祐 B09705009

2022-10-10

1 Environment

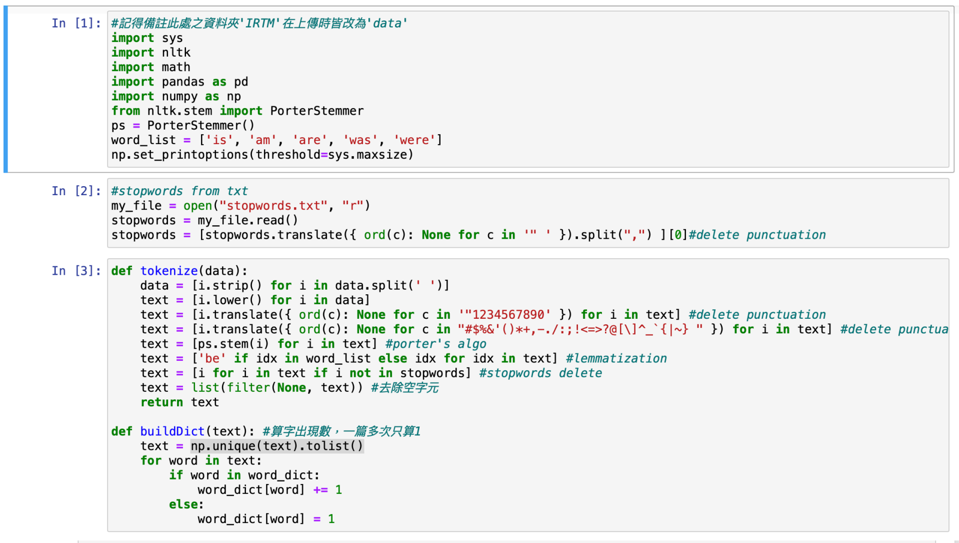
Using Jupyter Noteook

2 Language

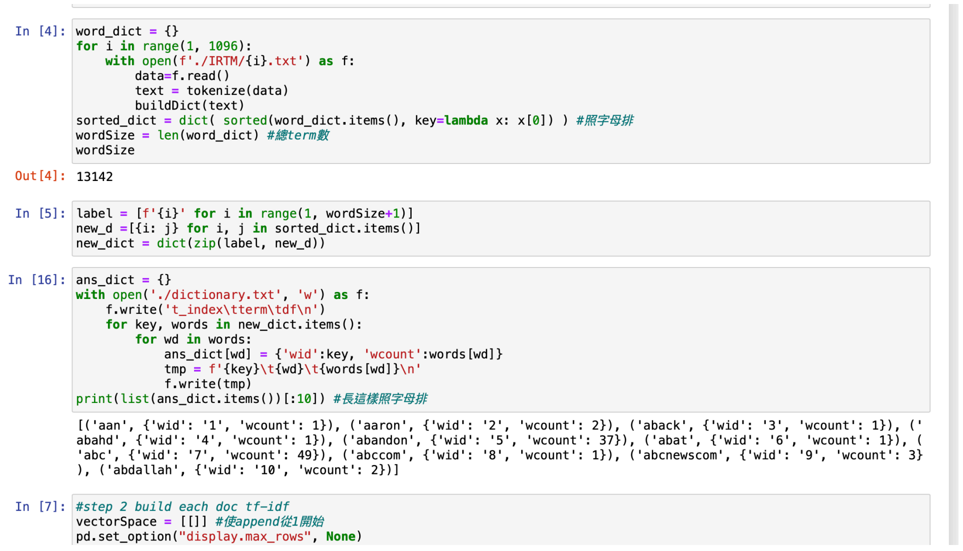
Python3

3 Execute

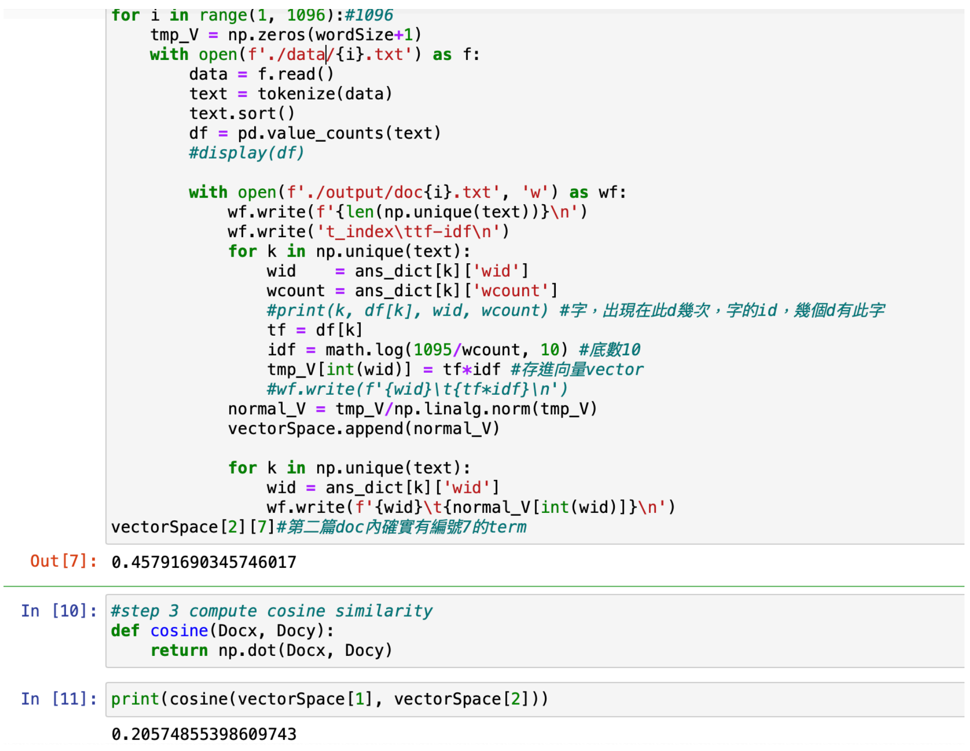
import sys, nltk, math, pandas, numpy, from nltk.stem import PorterStemmer
-> Use jupyter notebook to run the ipynb file.



```
In [1]: #記得備註此處之資料夾'IRTM'在上傳時皆改為'data'
        import sys
        import nltk
        import math
        import pandas as pd
        import numpy as np
        from nltk.stem import PorterStemmer
        ps = PorterStemmer()
        word_list = ['is', 'am', 'are', 'was', 'were']
        np.set_printoptions(threshold=sys.maxsize)

In [2]: #stopwords from txt
        my_file = open("stopwords.txt", "r")
        stopwords = my_file.read()
        stopwords = [stopwords.translate({ ord(c): None for c in '" ' }).split(",") ][0]#delete punctuation

In [3]: def tokenize(data):
            data = [i.strip() for i in data.split(' ')]
            text = [i.lower() for i in data]
            text = [i.translate({ ord(c): None for c in '"1234567890' }) for i in text] #delete punctuation
            text = [i.translate({ ord(c): None for c in "#$%&'()*+,-./:;!<=>?@[\]^_`{|~} " }) for i in text] #delete punctua
            text = [ps.stem(i) for i in text] #porter's algo
            text = ['be' if idx in word_list else idx for idx in text] #lemmatization
            text = [i for i in text if i not in stopwords] #stopwords delete
            text = list(filter(None, text)) #去除空字元
            return text

        def buildDict(text): #算字出現數，一篇多次只算1
            text = np.unique(text).tolist()
            for word in text:
                if word in word_dict:
                    word_dict[word] += 1
                else:
                    word_dict[word] = 1
```



```
In [4]: word_dict = {}
        for i in range(1, 1096):
            with open(f'./IRTM/{i}.txt') as f:
                data=f.read()
                text = tokenize(data)
                buildDict(text)
        sorted_dict = dict( sorted(word_dict.items(), key=lambda x: x[0]) ) #照字母排
        wordSize = len(word_dict) #總term數
        wordSize

Out[4]: 13142

In [5]: label = [f'{i}' for i in range(1, wordSize+1)]
        new_d =[{i: j} for i, j in sorted_dict.items()]
        new_dict = dict(zip(label, new_d))

In [16]: ans_dict = {}
         with open('./dictionary.txt', 'w') as f:
             f.write('t_index\tterm\tdf\n')
             for key, words in new_dict.items():
                 for wd in words:
                     ans_dict[wd] = {'wid':key, 'wcount':words[wd]}
                     tmp = f'{key}\t{wd}\t{words[wd]}\n'
                     f.write(tmp)
         print(list(ans_dict.items())[:10]) #長這樣照字母排

         [('aan', {'wid': '1', 'wcount': 1}), ('aaron', {'wid': '2', 'wcount': 2}), ('aback', {'wid': '3', 'wcount': 1}), ('
         abahd', {'wid': '4', 'wcount': 1}), ('abandon', {'wid': '5', 'wcount': 37}), ('abat', {'wid': '6', 'wcount': 1}), (
         'abc', {'wid': '7', 'wcount': 49}), ('abccom', {'wid': '8', 'wcount': 1}), ('abcnewscom', {'wid': '9', 'wcount': 3}
         ), ('abdallah', {'wid': '10', 'wcount': 2})]

In [7]: #step 2 build each doc tf-idf
        vectorSpace = [[]] #使append從1開始
        pd.set_option("display.max_rows", None)
```



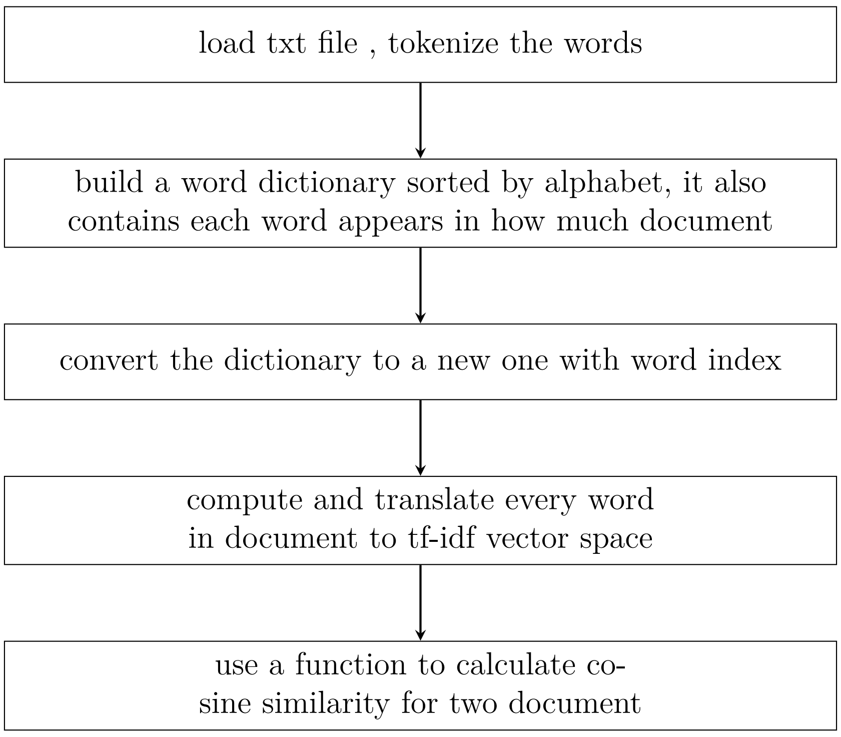
```
for i in range(1, 1096):#1096
    tmp_V = np.zeros(wordSize+1)
    with open(f'./data/{i}.txt') as f:
        data = f.read()
        text = tokenize(data)
        text.sort()
        df = pd.value_counts(text)
        #display(df)

        with open(f'./output/doc{i}.txt', 'w') as wf:
            wf.write(f'{len(np.unique(text))}\n')
            wf.write('t_index\ttf-idf\n')
            for k in np.unique(text):
                wid    = ans_dict[k]['wid']
                wcount = ans_dict[k]['wcount']
                #print(k, df[k], wid, wcount) #字，出現在此d幾次，字的id，幾個d有此字
                tf = df[k]
                idf = math.log(1095/wcount, 10) #底數10
                tmp_V[int(wid)] = tf*idf #存進向量vector
                #wf.write(f'{wid}\t{tf*idf}\n')
            normal_V = tmp_V/np.linalg.norm(tmp_V)
            vectorSpace.append(normal_V)

            for k in np.unique(text):
                wid = ans_dict[k]['wid']
                wf.write(f'{wid}\t{normal_V[int(wid)]}\n')
vectorSpace[2][7]#第二篇doc內確實有編號7的term
```

```
Out[7]: 0.45791690345746017
```

```
In [10]: #step 3 compute cosine similarity
         def cosine(Docx, Docy):
             return np.dot(Docx, Docy)
```

```
In [11]: print(cosine(vectorSpace[1], vectorSpace[2]))

         0.20574855398609743
```

4 Program Logic


load txt file , tokenize the words
build a word dictionary sorted by alphabet, it also contains each word appears in how much document
convert the dictionary to a new one with word index
compute and translate every word in document to tf-idf vector space
use a function to calculate cosine similarity for two document