資訊檢索與文字探勘導論
Homework 3

資管三吳驊祐 B09705009

2022-11-27

1 Environment

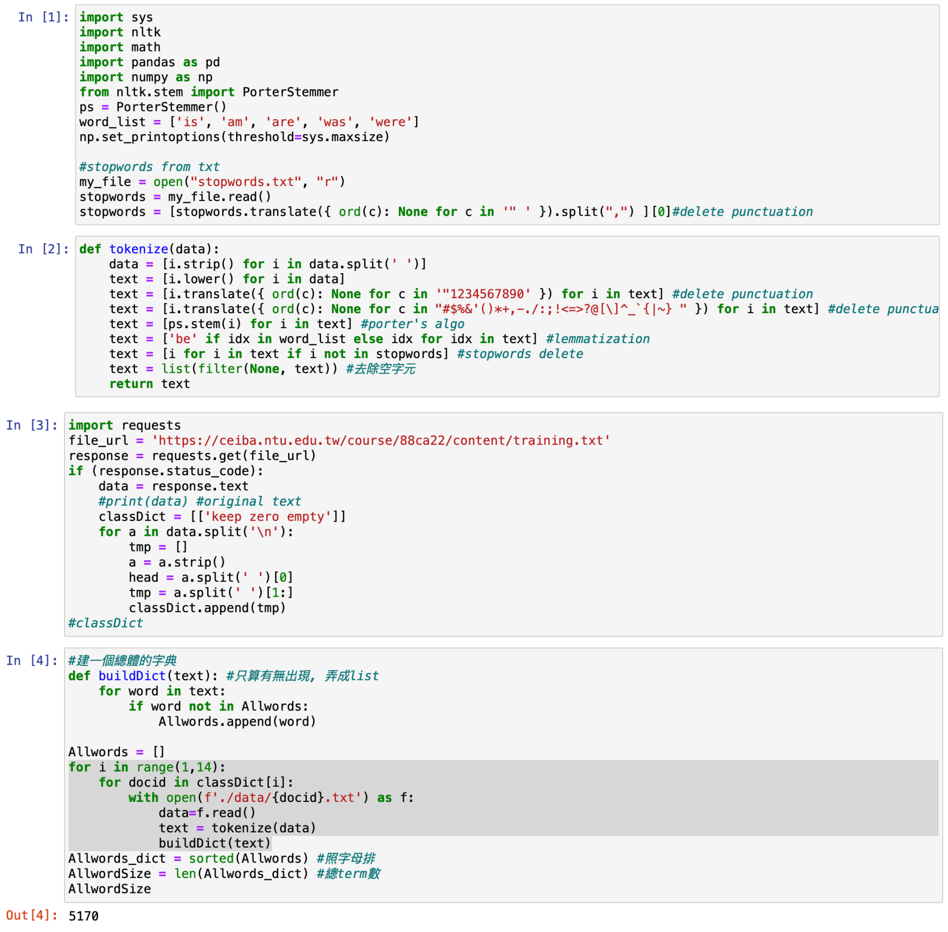
Using Jupyter Noteook

2 Language

Python3

3 Execute

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



```
In [1]: 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)

        #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 [2]: 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
```

```
In [3]: import requests
        file_url = 'https://ceiba.ntu.edu.tw/course/88ca22/content/training.txt'
        response = requests.get(file_url)
        if (response.status_code):
            data = response.text
            #print(data) #original text
            classDict = [['keep zero empty']]
            for a in data.split('\n'):
                tmp = []
                a = a.strip()
                head = a.split(' ')[0]
                tmp = a.split(' ')[1:]
                classDict.append(tmp)
        #classDict
```

```
In [4]: #建一個總體的字典
        def buildDict(text): #只算有無出現, 弄成list
            for word in text:
                if word not in Allwords:
                    Allwords.append(word)

        Allwords = []
        for i in range(1,14):
            for docid in classDict[i]:
                with open(f'./data/{docid}.txt') as f:
                    data=f.read()
                    text = tokenize(data)
                    buildDict(text)
        Allwords_dict = sorted(Allwords) #照字母排
        AllwordSize = len(Allwords_dict) #總term數
        AllwordSize

Out[4]: 5170
```

```
In [5]: eachCategorycnt_dict = [['keep zero empty']] #方便計算從1開始到13
        wcount_dict = [['keep zero empty']] #計算每類別中'總字數'有多少
        for i in range(1,14):
            dict_from_list = {k: 0 for k in Allwords_dict}
            wcount = 0
            for docid in classDict[i]:
                with open(f'./data/{docid}.txt') as f:
                    data=f.read()
                    text = tokenize(data)
                    for word in text:
                        dict_from_list[word] += 1
                        wcount+=1
            eachCategorycnt_dict.append(dict_from_list)
            wcount_dict.append(wcount)
        #eachCategorycnt_dict
```

```
In [6]: #算出每個字在每個類別中出現的比率分數
        import copy

        params_dict = copy.deepcopy(eachCategorycnt_dict)
        for category in range(1,14):
            for k in params_dict[category]:
                params_dict[category][k] = (params_dict[category][k]+1) / (wcount_dict[category]+AllwordSize)
        #params_dict[13]
```

Feature selection

```
In [7]: #先建立一個算所有terms在所有文章中出現次數加總
        #先前已計算個類別中字典裡所有字出現次數在eachCategorycnt_dict
        #其中可能有反指標
        wordSum_dict = {k: 0 for k in Allwords_dict}
        for wd in wordSum_dict:
            for i in range(1,14):
                wordSum_dict[wd] += eachCategorycnt_dict[i][wd] #每個class出現次數加總即為總出現次數

        chi_score_dict = {k: 0 for k in Allwords_dict} #另創一個紀錄chi-square value
        for wd in wordSum_dict:
            expected = wordSum_dict[wd] / 13
            score = 0
            for c in range(1,14):
                tmp = ((eachCategorycnt_dict[i][wd]-expected)**2)/expected
                score += tmp
            chi_score_dict[wd] = score

        chi_score_dict = dict( sorted(chi_score_dict.items(), key=lambda x: x[1], reverse=True) ) #照大小排
        #chi_score_dict
```

```
In [8]: imp_feature = []
        for key in chi_score_dict.keys():
            imp_feature.append(key)
        imp_feature = imp_feature[545:1035]
        len(imp_feature)

Out[8]: 490
```

```
In [9]: test_id = np.arange(1,1096)
        test_id = [x for x in test_id if x not in [int(item) for sublist in classDict[1:] for item in sublist]] #用來test的
```

```
In [10]: def computeCat(docid): #return category
             with open(f'./data/{docid}.txt') as f:
                 data=f.read()
                 text = tokenize(data)
                 text = [x for x in text if x in imp_feature]
                 maxC = 0
                 maxValue = -2e200
                 for i in range(1, 14):
                     tmpValue = 0
                     for word in text:
                         tmpValue += math.log(params_dict[i][word])
                     if tmpValue > maxValue:
                         maxValue = tmpValue
                         maxC = i
                 return maxC
```

```
In [11]: data = [] #用來最後輸出
         for tid in test_id:
             tid_category = computeCat(tid)
             data.append([tid, tid_category])
         df = pd.DataFrame(data, columns=['Id', 'Value'])
         df

Out[11]:
```

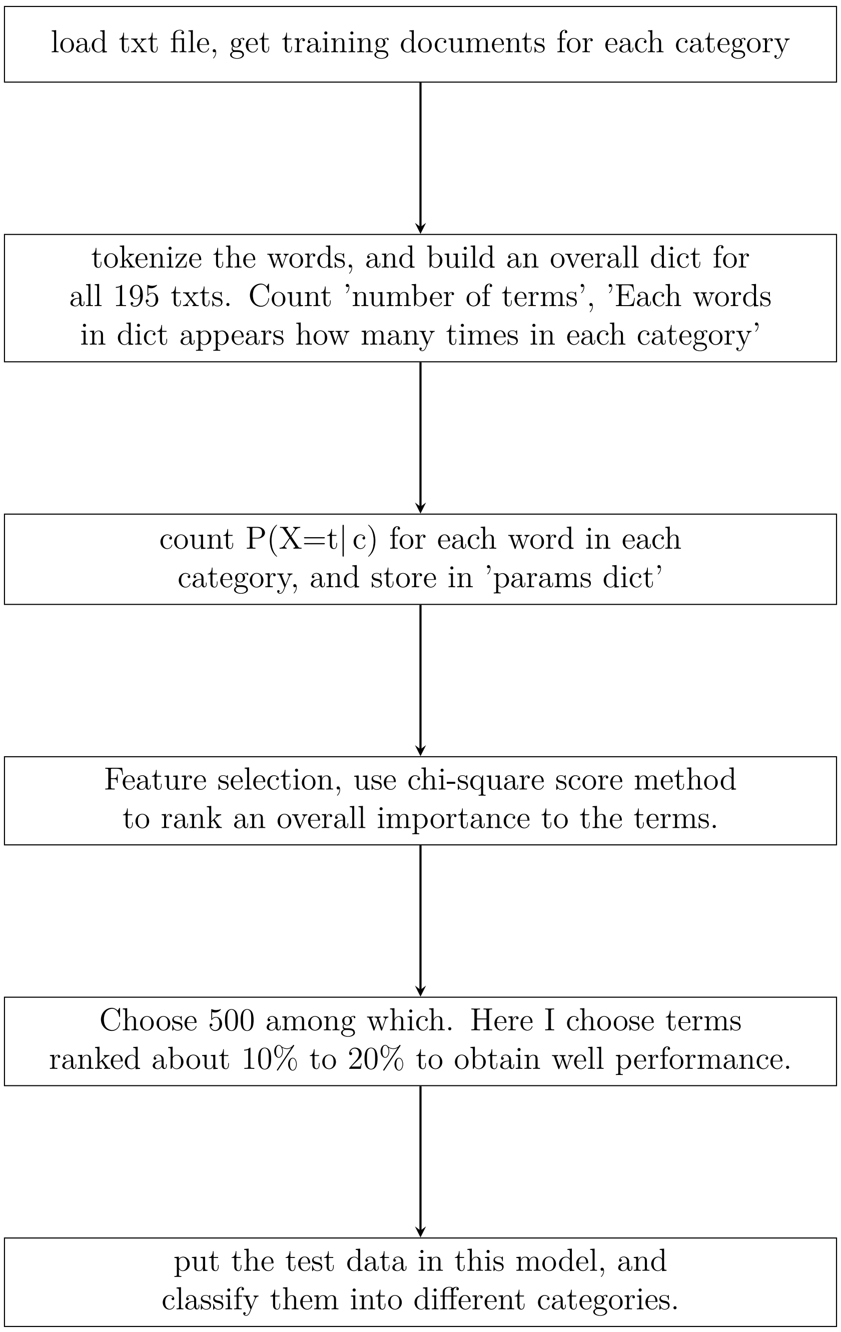
	Id	Value
0	17	2
1	18	2

```
In [12]: df['Value'].value_counts()

Out[12]: 9     277
         2     112
         11     69
         5      68
         7      55
         6      49
         12     46
         8      42
         3      40
         4      40
         1      39
         10     36
         13     27
         Name: Value, dtype: int64
```

```
In [13]: #df2 = pd.DataFrame(params_dict[13].items(), columns=['wd', 'val'])
         #df2.sort_values(by='val')
         import os
         df.to_csv('./outAns.csv', index=False)
```

4 Program Logic


load txt file, get training documents for each category
tokenize the words, and build an overall dict for all 195 txts. Count 'number of terms', 'Each words in dict appears how many times in each category'
count P(X=t|c) for each word in each category, and store in 'params dict'
Feature selection, use chi-square score method to rank an overall importance to the terms.
Choose 500 among which. Here I choose terms ranked about 10% to 20% to obtain well performance.
put the test data in this model, and classify them into different categories.