Information Retrieval and Text Mining: Homework #3

R08725008 資管碩二 周若涓

執行環境

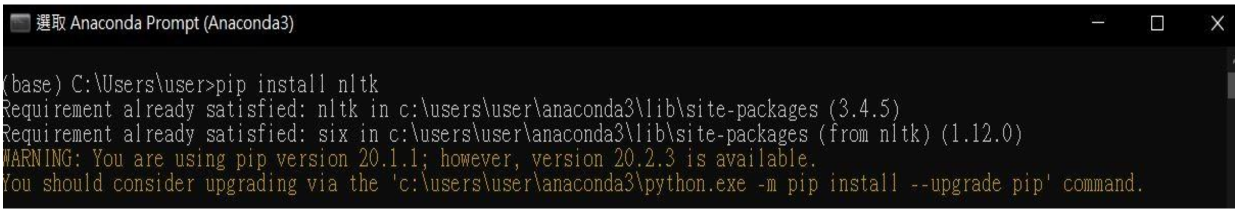
- Jupyter Notebook

程式語言

- Python 3

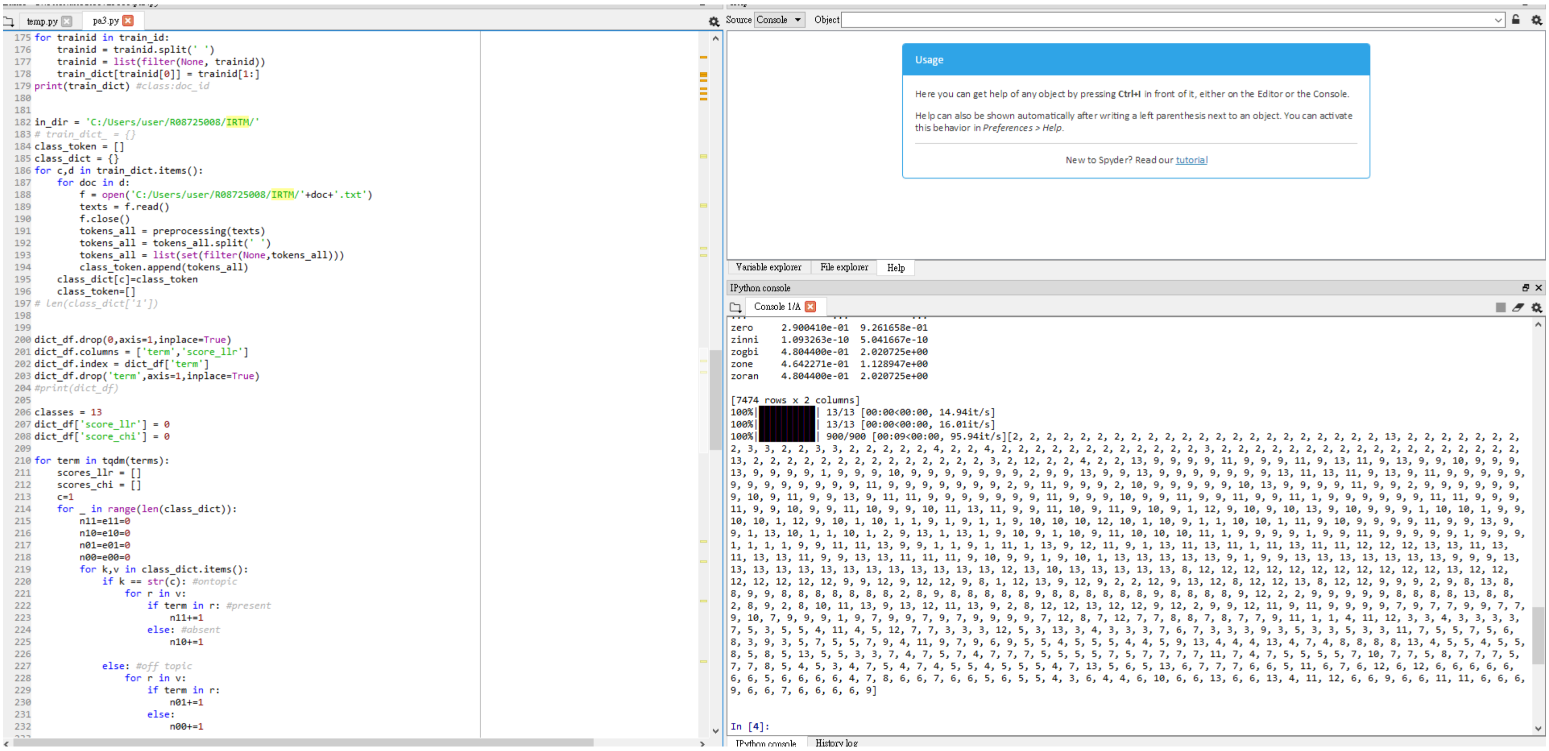
執行方式

- 在執行前需安裝 nltk 套件 (command: pip install nltk)



```
(base) C:\Users\user>pip install nltk
Requirement already satisfied: nltk in c:\users\user\anaconda3\lib\site-packages (3.4.5)
Requirement already satisfied: six in c:\users\user\anaconda3\lib\site-packages (from nltk) (1.12.0)
WARNING: You are using pip version 20.1.1; however, version 20.2.3 is available.
You should consider upgrading via the 'c:\users\user\anaconda3\python.exe -m pip install --upgrade pip' command.
```

- 安裝 pandas 套件 (command: pip install pandas)
- 安裝 numpy 套件 (command: pip install numpy)
- 執行環境:
  - 可以利用 Spyder 開啟 pa3.py，並執行



```
for trainid in train_id:
    trainid = trainid.split(' ')
    trainid = list(filter(None, trainid))
    train_dict[trainid[0]] = trainid[1:]
print(train_dict) #class:doc_id


in_dir = 'C:/Users/user/R08725008/IRTM/'
# train_dict_ = {}
class_token = []
class_dict = {}
for c,d in train_dict.items():
    for doc in d:
        f = open('C:/Users/user/R08725008/IRTM/'+doc+'.txt')
        texts = f.read()
        f.close()
        tokens_all = preprocessing(texts)
        tokens_all = tokens_all.split(' ')
        tokens_all = list(set(filter(None,tokens_all)))
        class_token.append(tokens_all)
    class_dict[c]=class_token
    class_token=[]
# len(class_dict['1'])


dict_df.drop(0,axis=1,inplace=True)
dict_df.columns = ['term','score_llr']
dict_df.index = dict_df['term']
dict_df.drop('term',axis=1,inplace=True)
#print(dict_df)

classes = 13
dict_df['score_llr'] = 0
dict_df['score_chi'] = 0

for term in tqdm(terms):
    scores_llr = []
    scores_chi = []
    c=1
    for _ in range(len(class_dict)):
        n11=e11=0
        n10=e10=0
        n01=e01=0
        n00=e00=0
        for k,v in class_dict.items():
            if k == str(c): #ontopic
                for r in v:
                    if term in r: #present
                        n11+=1
                    else: #absent
                        n10+=1

            else: #off topic
                for r in v:
                    if term in r:
                        n01+=1
                    else:
                        n00+=1
```

Usage

Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

```
zero     2.900410e-01  9.261658e-01
zinni    1.093263e-10  5.041667e-10
zogbi    4.804400e-01  2.020725e+00
zone     4.642271e-01  1.128947e+00
zoran    4.804400e-01  2.020725e+00

[7474 rows x 2 columns]
100%|██████████| 13/13 [00:00<00:00, 14.94it/s]
100%|██████████| 13/13 [00:00<00:00, 16.01it/s]
100%|██████████| 900/900 [00:09<00:00, 95.94it/s][2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 13, 2, 2, 2, 2, 2, 2, 2,
2, 3, 3, 2, 2, 3, 3, 2, 2, 2, 2, 2, 4, 2, 2, 4, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
13, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 3, 2, 12, 2, 2, 4, 2, 2, 13, 9, 9, 9, 9, 11, 9, 9, 9, 11, 9, 13, 11, 9, 13, 9, 9, 10, 9, 9, 9,
13, 9, 9, 9, 9, 1, 9, 9, 9, 10, 9, 9, 9, 9, 9, 9, 9, 2, 9, 9, 13, 9, 9, 13, 9, 9, 9, 9, 9, 9, 9, 13, 11, 13, 11, 9, 13, 9, 11, 9, 9, 9, 9, 9,
9, 9, 9, 9, 9, 9, 9, 9, 11, 9, 9, 9, 9, 9, 9, 9, 2, 9, 11, 9, 9, 9, 2, 10, 9, 9, 9, 9, 9, 10, 13, 9, 9, 9, 9, 11, 9, 9, 2, 9, 9, 9, 9, 9, 9,
9, 10, 9, 11, 9, 9, 13, 9, 11, 11, 9, 9, 9, 9, 9, 9, 9, 11, 9, 9, 9, 10, 9, 9, 11, 9, 9, 11, 9, 9, 11, 1, 9, 9, 9, 9, 9, 9, 11, 11, 9, 9, 9,
11, 9, 9, 10, 9, 9, 11, 10, 9, 9, 10, 11, 13, 11, 9, 9, 11, 10, 9, 11, 9, 10, 9, 1, 12, 9, 10, 9, 10, 13, 9, 10, 9, 9, 9, 1, 10, 10, 1, 9, 9,
10, 10, 1, 12, 9, 10, 1, 10, 1, 1, 9, 1, 9, 1, 1, 9, 10, 10, 10, 12, 10, 1, 10, 9, 1, 1, 10, 10, 1, 11, 9, 10, 9, 9, 9, 9, 11, 9, 9, 13, 9,
9, 1, 13, 10, 1, 1, 10, 1, 2, 9, 13, 1, 13, 1, 9, 10, 9, 1, 10, 9, 11, 10, 10, 10, 11, 1, 9, 9, 9, 9, 1, 9, 9, 11, 9, 9, 9, 9, 9, 1, 9, 9, 9,
1, 1, 1, 1, 9, 9, 11, 11, 13, 9, 9, 1, 1, 9, 1, 11, 1, 13, 9, 12, 11, 9, 1, 13, 11, 13, 11, 1, 11, 13, 11, 11, 12, 12, 12, 13, 13, 11, 13,
11, 13, 13, 11, 9, 9, 13, 13, 11, 11, 11, 9, 10, 9, 9, 1, 9, 10, 1, 13, 13, 13, 13, 13, 9, 1, 9, 9, 13, 13, 13, 13, 13, 13, 13, 9, 9, 9, 13,
13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 13, 12, 13, 10, 13, 13, 13, 13, 13, 8, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 12, 13, 12, 12,
12, 12, 12, 12, 12, 9, 9, 12, 9, 12, 12, 9, 8, 1, 12, 13, 9, 12, 9, 2, 2, 12, 9, 13, 12, 8, 12, 12, 13, 8, 12, 12, 9, 9, 9, 2, 9, 8, 13, 8,
8, 9, 9, 8, 8, 8, 8, 8, 8, 8, 2, 8, 9, 8, 8, 8, 8, 8, 9, 8, 8, 8, 8, 8, 8, 9, 8, 8, 8, 8, 9, 12, 2, 2, 9, 9, 9, 9, 9, 8, 8, 8, 8, 13, 8, 8,
2, 8, 9, 2, 8, 10, 11, 13, 9, 13, 12, 11, 13, 9, 2, 8, 12, 12, 13, 12, 12, 9, 12, 2, 9, 9, 12, 11, 9, 11, 9, 9, 9, 9, 7, 9, 7, 7, 9, 9, 7, 7,
9, 10, 7, 9, 9, 9, 1, 9, 7, 9, 9, 7, 9, 7, 9, 9, 9, 9, 7, 12, 8, 7, 12, 7, 7, 8, 8, 7, 8, 7, 7, 9, 11, 1, 1, 4, 11, 12, 3, 3, 4, 3, 3, 3, 3,
7, 5, 3, 5, 5, 4, 11, 4, 5, 12, 7, 7, 3, 3, 3, 12, 5, 3, 13, 3, 4, 3, 3, 3, 7, 6, 7, 3, 3, 3, 9, 3, 5, 3, 3, 5, 3, 3, 11, 7, 5, 5, 7, 5, 6,
8, 3, 9, 3, 5, 7, 5, 5, 7, 9, 4, 11, 9, 7, 9, 6, 9, 5, 5, 4, 5, 5, 5, 4, 4, 5, 9, 13, 4, 4, 4, 13, 4, 7, 4, 8, 8, 8, 8, 13, 4, 5, 5, 4, 5, 5,
8, 5, 8, 5, 13, 5, 5, 3, 3, 7, 4, 7, 5, 7, 4, 7, 7, 7, 5, 5, 5, 5, 7, 5, 7, 7, 7, 7, 11, 7, 4, 7, 5, 5, 5, 5, 7, 10, 7, 7, 5, 8, 7, 7, 7, 5,
7, 7, 8, 5, 4, 5, 3, 4, 7, 5, 4, 7, 4, 5, 5, 4, 5, 5, 5, 4, 7, 13, 5, 6, 5, 13, 6, 7, 7, 7, 6, 6, 5, 11, 6, 7, 6, 12, 6, 12, 6, 6, 6, 6, 6,
6, 6, 5, 6, 6, 6, 6, 4, 7, 8, 6, 6, 7, 6, 6, 5, 6, 5, 5, 4, 3, 6, 4, 4, 6, 10, 6, 6, 13, 6, 6, 13, 4, 11, 12, 6, 6, 9, 6, 6, 11, 11, 6, 6, 6,
9, 6, 6, 7, 6, 6, 6, 6, 9]

In [4]:
```

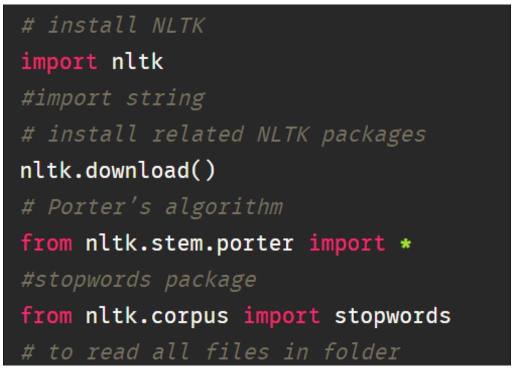
- 確保提供的 IRTM 資料夾放於 C:/Users/user/R08725008/ 目錄下
- 確保提供的 training.txt 放於 C:/Users/user/R08725008/ 目錄下
- 產出的 res.csv 預設放於 C:/Users/user/R08725008/ 目錄下

作業邏輯說明

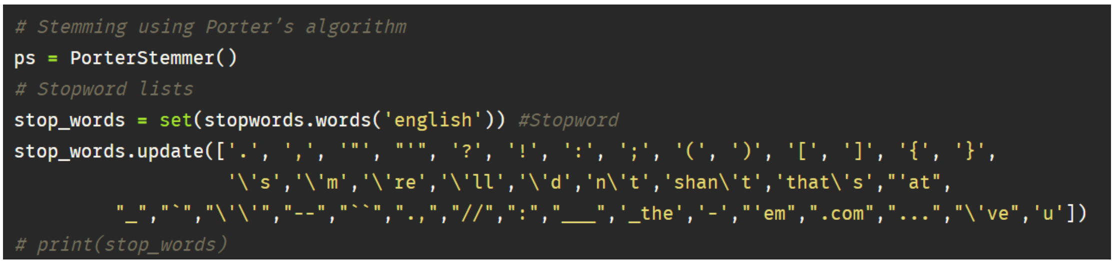
Part 1: Preprocessing

1.

- 利用 nltk 套件初始化 PorterStemmer
- import stopwords 套件，並增加 list，其為觀察到可能為 stopwords



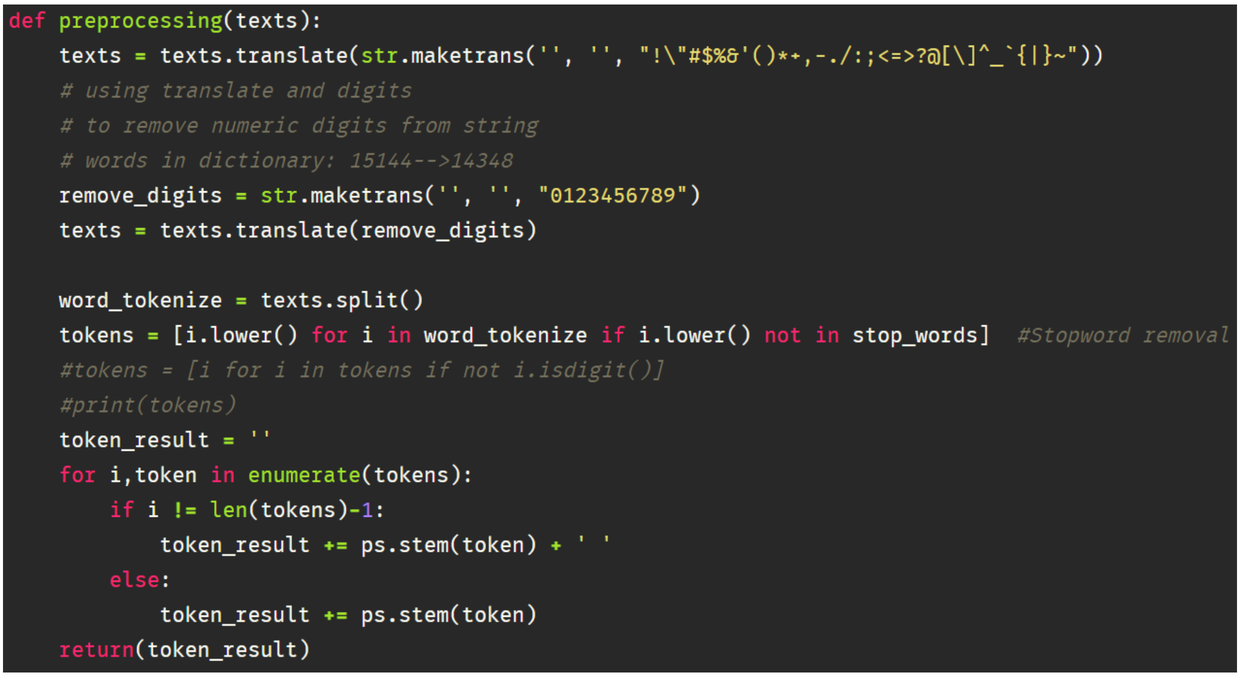
```
# install NLTK
import nltk
#import string
# install related NLTK packages
nltk.download()
# Porter’s algorithm
from nltk.stem.porter import *
#stopwords package
from nltk.corpus import stopwords
# to read all files in folder
```



```
# Stemming using Porter’s algorithm
ps = PorterStemmer()
# Stopword lists
stop_words = set(stopwords.words('english')) #Stopword
stop_words.update(['.', ',', '"', "'", '?', '!', ':', ';', '(', ')', '[', ']', '{', '}',
                   '\'s','\'m','\'re','\'ll','\'d','n\'t','shan\'t','that\'s',"'at",
        "_","`","\'\'","--","``",".,","//",":","___",'_the','-',"'em",".com","...","\'ve",'u'])
# print(stop_words)
```

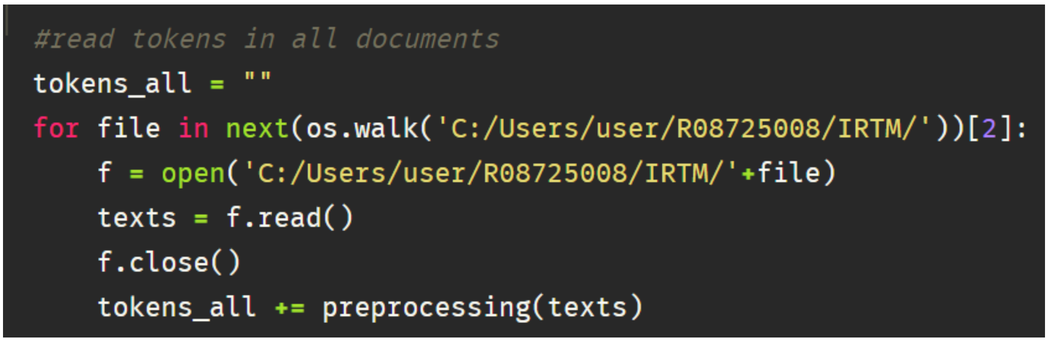
2. 定義 preprocessing function，有以下功能

- 去除 punctuation
- 加上去除數字的處理
- Tokenize
- 將讀入的文件轉換為小寫，如果不在 stop words 當中的 word 才會保留
- 最後將每個 token 進行 stemming，並加回字串當中
- 把結果輸出 token_result



```
def preprocessing(texts):
    texts = texts.translate(str.maketrans('', '', "!\"#$%&'()*+,-./:;<=>?@[\]^_`{|}~"))
    # using translate and digits
    # to remove numeric digits from string
    # words in dictionary: 15144-->14348
    remove_digits = str.maketrans('', '', "0123456789")
    texts = texts.translate(remove_digits)

    word_tokenize = texts.split()
    tokens = [i.lower() for i in word_tokenize if i.lower() not in stop_words]  #Stopword removal
    #tokens = [i for i in tokens if not i.isdigit()]
    #print(tokens)
    token_result = ''
    for i,token in enumerate(tokens):
        if i != len(tokens)-1:
            token_result += ps.stem(token) + ' '
        else:
            token_result += ps.stem(token)
    return(token_result)
```

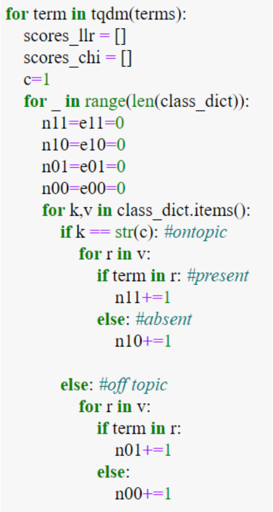
3. 讀入所有資料夾中的 documents，經過前處理後的所有 terms 收集至 tokens_all



```
#read tokens in all documents
tokens_all = ""
for file in next(os.walk('C:/Users/user/R08725008/IRTM/'))[2]:
    f = open('C:/Users/user/R08725008/IRTM/'+file)
    texts = f.read()
    f.close()
    tokens_all += preprocessing(texts)
```

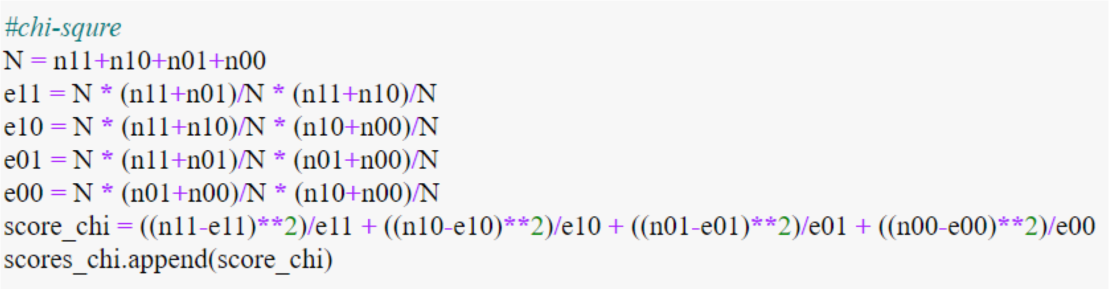
Part 2: Feature selection

1. 讀取 dictionary.txt 中所有的 terms，並建立 term list，計算每個 term 是否出現在各個 class 的 training doc 中
    - Term-class matrix 計算的結果可用於 chi-square test 與 log likelihood ratios



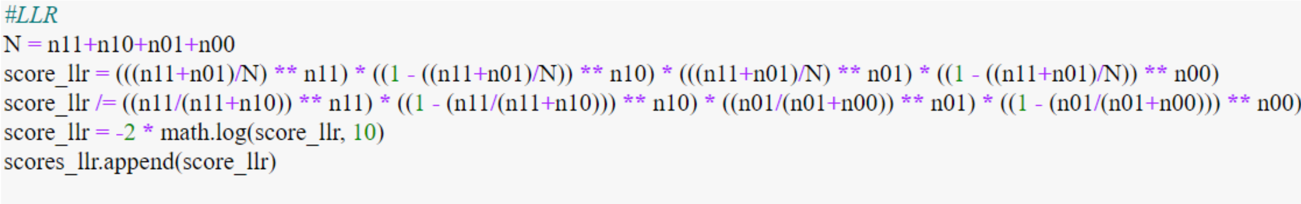
```
for term in tqdm(terms):
    scores_llr = []
    scores_chi = []
    c=1
    for _ in range(len(class_dict)):
        n11=e11=0
        n10=e10=0
        n01=e01=0
        n00=e00=0
        for k,v in class_dict.items():
            if k == str(c): #ontopic
                for r in v:
                    if term in r: #present
                        n11+=1
                    else: #absent
                        n10+=1

            else: #off topic
                for r in v:
                    if term in r:
                        n01+=1
                    else:
                        n00+=1
```

2. 根據講義上的公式，實作 chi-square test，觀察 count E 與 觀察值 N 是否彼此獨立，字與文章越獨立計算出來的數值越大



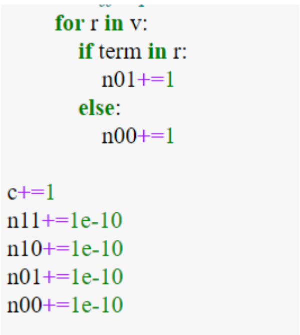
```
#chi-squre
N = n11+n10+n01+n00
e11 = N * (n11+n01)/N * (n11+n10)/N
e10 = N * (n11+n10)/N * (n10+n00)/N
e01 = N * (n11+n01)/N * (n01+n00)/N
e00 = N * (n01+n00)/N * (n10+n00)/N
score_chi = ((n11-e11)**2)/e11 + ((n10-e10)**2)/e10 + ((n01-e01)**2)/e01 + ((n00-e00)**2)/e00
scores_chi.append(score_chi)
```

3. 根據講義上的公式，實作 log likelihood ratios，計算 h1 與 h2 的比值，字與文章越獨立計算出來的數值越大



```
#LLR
N = n11+n10+n01+n00
score_llr = (((n11+n01)/N) ** n11) * ((1 - ((n11+n01)/N)) ** n10) * (((n11+n01)/N) ** n01) * ((1 - ((n11+n01)/N)) ** n00)
score_llr /= ((n11/(n11+n10)) ** n11) * ((1 - (n11/(n11+n10))) ** n10) * ((n01/(n01+n00)) ** n01) * ((1 - (n01/(n01+n00))) ** n00)
score_llr = -2 * math.log(score_llr, 10)
scores_llr.append(score_llr)
```

4. 在計算的過程中遇到 *float division by zero python* 的 error message，因此 n11, n10, n01, n00，都會加上一個非常小的數值

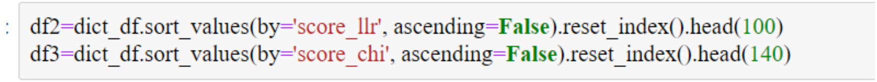


```
        for r in v:
            if term in r:
                n01+=1
            else:
                n00+=1

c+=1
n11+=1e-10
n10+=1e-10
n01+=1e-10
n00+=1e-10
```

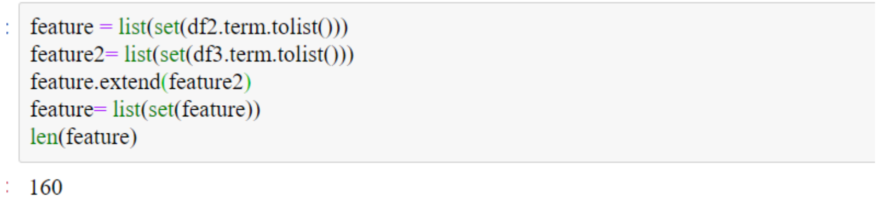
5. 經過各種嘗試發現 chi-square test 挑出的字似乎比 log likelihood ratios 強，因此，將 dataframe df2 用 log likelihood ratios 的值排大小，dataframe df3

用 chi-square test 的值排大小，接著 chi-square test 挑前 140 個 terms，log likelihood ratios 挑前 100 個 terms

- 將上述兩個 dataframe 的 term field 取出，做成 list，在把重複取的 terms 去掉，最後留下 160 個 terms
- 透過調節選取的 terms 發現，選 160 個 terms 在目前的實作過程為最佳解，只要再選越多 terms，出來的 f1-score 越低



```
df2=dict_df.sort_values(by='score_llr', ascending=False).reset_index().head(100)
df3=dict_df.sort_values(by='score_chi', ascending=False).reset_index().head(140)
```



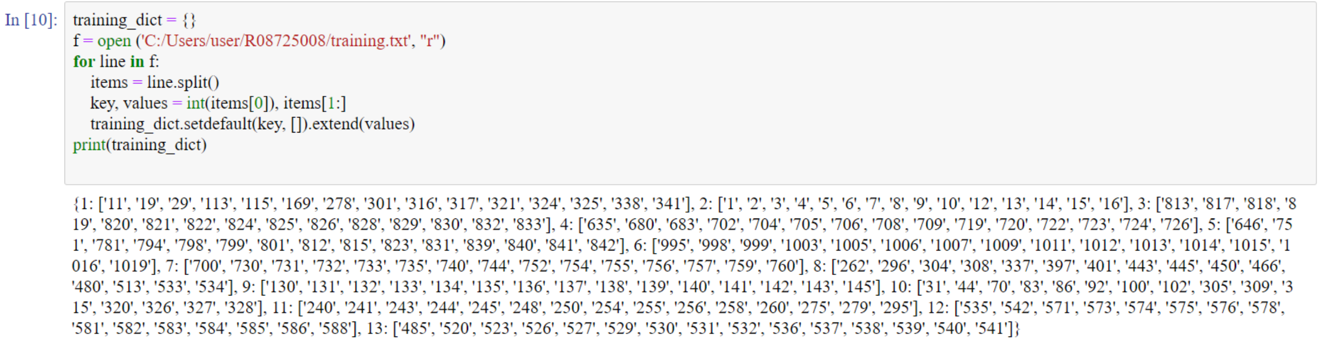
```
feature = list(set(df2.term.tolist()))
feature2= list(set(df3.term.tolist()))
feature.extend(feature2)
feature= list(set(feature))
len(feature)
```

```
160
```

Part 3: Multinomial Model for Text Classification

1. Training

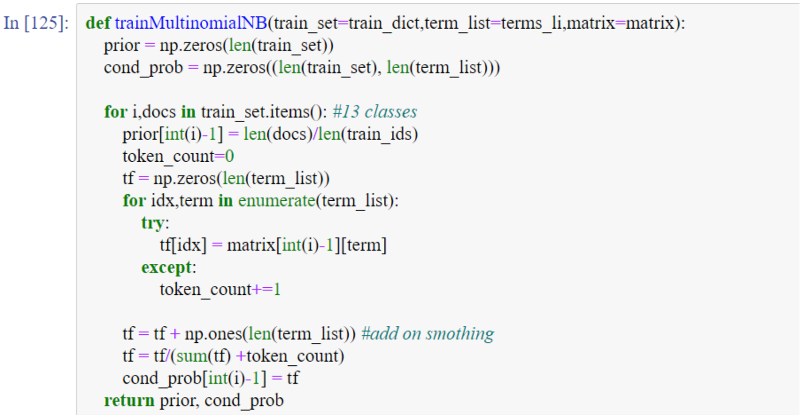
- 讀入 training.txt，並用 dictionary 儲存 class 與其對應類別的文章



```
In [10]: training_dict = {}
         f = open ('C:/Users/user/R08725008/training.txt', "r")
         for line in f:
             items = line.split()
             key, values = int(items[0]), items[1:]
             training_dict.setdefault(key, []).extend(values)
         print(training_dict)
```

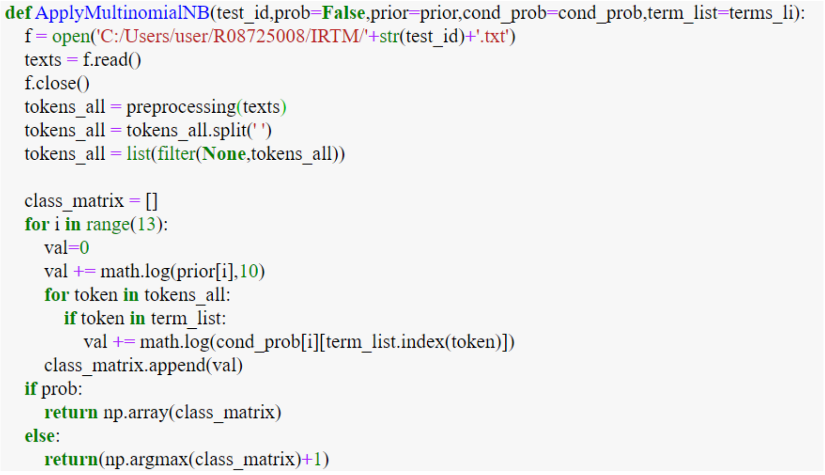
```
{1: ['11', '19', '29', '113', '115', '169', '278', '301', '316', '317', '321', '324', '325', '338', '341'], 2: ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '12', '13', '14', '15', '16'], 3: ['813', '817', '818', '8
19', '820', '821', '822', '824', '825', '826', '828', '829', '830', '832', '833'], 4: ['635', '680', '683', '702', '704', '705', '706', '708', '709', '719', '720', '722', '723', '724', '726'], 5: ['646', '75
1', '781', '794', '798', '799', '801', '812', '815', '823', '831', '839', '840', '841', '842'], 6: ['995', '998', '999', '1003', '1005', '1006', '1007', '1009', '1011', '1012', '1013', '1014', '1015', '1
016', '1019'], 7: ['700', '730', '731', '732', '733', '735', '740', '744', '752', '754', '755', '756', '757', '759', '760'], 8: ['262', '296', '304', '308', '337', '397', '401', '443', '445', '450', '466',
'480', '513', '533', '534'], 9: ['130', '131', '132', '133', '134', '135', '136', '137', '138', '139', '140', '141', '142', '143', '145'], 10: ['31', '44', '70', '83', '86', '92', '100', '102', '305', '309', '3
15', '320', '326', '327', '328'], 11: ['240', '241', '243', '244', '245', '248', '250', '254', '255', '256', '258', '260', '275', '279', '295'], 12: ['535', '542', '571', '573', '574', '575', '576', '578',
'581', '582', '583', '584', '585', '586', '588'], 13: ['485', '520', '523', '526', '527', '529', '530', '531', '532', '536', '537', '538', '539', '540', '541']}
```

- 計算 prior matrix，training docs 中各個 class 占比為何
- 計算 cond_prob，所有 class c 的 docs 共有幾個 terms，計算各個 term 出現次數在其中的占比
- 避免 zero probability problem，在此做 add one smoothing
- 最後得到一個 13*160 的 matrix
    - Row: 13，class 總數
    - Col: 160，feature selection 選出來的 terms



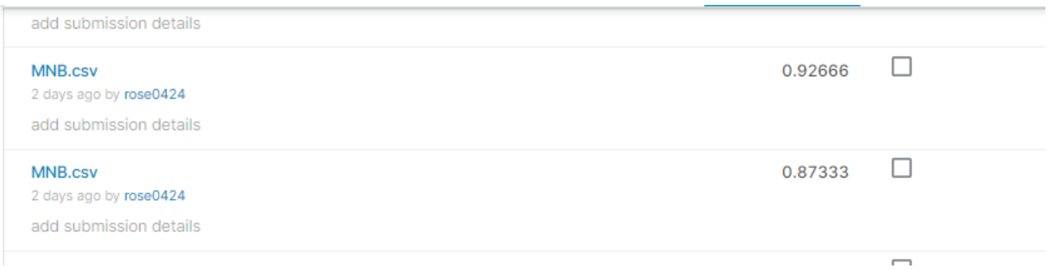
```
def trainMultinomialNB(train_set=train_dict,term_list=terms_li,matrix=matrix):
    prior = np.zeros(len(train_set))
    cond_prob = np.zeros((len(train_set), len(term_list)))

    for i,docs in train_set.items(): #13 classes
        prior[int(i)-1] = len(docs)/len(train_ids)
        token_count=0
        tf = np.zeros(len(term_list))
        for idx,term in enumerate(term_list):
            try:
                tf[idx] = matrix[int(i)-1][term]
            except:
                token_count+=1

        tf = tf + np.ones(len(term_list)) #add on smothing
        tf = tf/(sum(tf) +token_count)
        cond_prob[int(i)-1] = tf
    return prior, cond_prob
```

2. Testing
    - Matrix 中各個 class 的 score 先加上 prior 基本分
    - 讀入一篇文章(非 training doc)，經前處理留下的 terms，看其是否在 feature selection term set 中，如果在就在某個 class 的 entry 加上 13*160 的 matrix 的對應值
    - Socre-matrix 中最大值即為結果



```
def ApplyMultinomialNB(test_id,prob=False,prior=prior,cond_prob=cond_prob,term_list=terms_li):
    f = open('C:/Users/user/R08725008/IRTM/'+str(test_id)+'.txt')
    texts = f.read()
    f.close()
    tokens_all = preprocessing(texts)
    tokens_all = tokens_all.split(' ')
    tokens_all = list(filter(None,tokens_all))

    class_matrix = []
    for i in range(13):
        val=0
        val += math.log(prior[i],10)
        for token in tokens_all:
            if token in term_list:
                val += math.log(cond_prob[i][term_list.index(token)])
        class_matrix.append(val)
    if prob:
        return np.array(class_matrix)
    else:
        return(np.argmax(class_matrix)+1)
```

Part 4: Kaggle result



add submission details

MNB.csv 0.92666 □
2 days ago by rose0424
add submission details

MNB.csv 0.87333 □
2 days ago by rose0424
add submission details