1. 執行環境 & 作業系統

## 本次採用的作業系統為win 10 64位元，執行環境為Visual Studio Code，需要預先安裝[Natural Language Toolkit](http://www.nltk.org/) tool 。首先要先在電腦用cmd安裝各個tool的工具包

1. 程式語言, 版本

使用的程式語言和版本為python 3.7。

1. 執行方式

先將要處理的1095個文字檔案存到IRTM的資料夾，training data 的文件分類txt檔-classification.txt與python source code 存放在同一個資料夾，使用Visual Studio Code 編寫完後，cmd 執行python程式。

* 執行畫面

1. 作業處理邏輯說明

首先要 import 多項套件，例如:

|  |
| --- |
| import math  import string  from nltk.stem import PorterStemmer  from nltk.tokenize import WordPunctTokenizer  from nltk.tokenize import word\_tokenize  from nltk.tokenize import wordpunct\_tokenize  from nltk.corpus import stopwords  import nltk |

* 一開始先將training分類的文件編號取出，用dict\_xx={}紀錄文件id
* Feature selection方面，我採用的是frequency based作法，先將training data有的文件所包含的term中，取出前500個出現最多次的，視他們為features

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | |  | | --- | | zz=nltk.FreqDist(zz)  word\_features\_dict={}  # print(zz.most\_common(500)) #計算出最多出現的前500 個term  word\_features = list(zz.keys())[:500]  for i in range(len(word\_features)):  word\_features\_dict[word\_features[i]]=int(i) #word\_feature\_dict value為前500最多出現的Term | | |

1095個文件先經過處理，排除前500個以外的其他term

|  |
| --- |
| term\_word\_feature\_all=[]  for i in range(len(term)):  term\_word\_feature=[]  for j in range(len(term[i])):  if term[i][j] in word\_features\_dict: #保存前500多個term，其餘省略掉  term\_word\_feature.append(term[i][j])  term\_word\_feature\_all.append(term\_word\_feature)  # term\_word\_feature\_all為一個陣列，儲存每一個文件編號裡面包含的前500term |

* 計算每個term在training data 的prior probability，用word\_probability={}字典紀錄並存到class\_word\_probability=[]陣列，算式要經過smoothing

|  |
| --- |
| class\_word\_probability=[]  from collections import Counter  for i in range(len(class\_all\_term\_dict)):  word\_probability={} #輸入key “term” 就可以得到term 的 prior probability  x=Counter(class\_all\_term\_dict[i])  for word in x: word\_probability[word]=(x[word]+1)/(len(class\_all\_term\_dict[i])+500)  class\_word\_probability.append(word\_probability) |

1. 任何在此作業中的心得

這次作業中首先要選擇feature selection的問題，我採用的是**frequency based**的作法，一開始**沒注意training data和testing data的區別**，因此用了整整1095個文件去做feature extraction，好在後來觀念的釐清幫助修正了此問題；至於word probability 計算的公式部分，並沒有套用log 的方式去計算(因為考慮到log 可能會產生負值，對於我的程式邏輯會出現錯誤)；最後比較重大的問題是要判別testing data 裡面的文件分別**有無出現**某個term(在第k個class 所出現的term)，再套用word probability去做計算，否則會產生問題。我認為此次作業有助於使我更加了解訓練模型的基礎概念和步驟，也認識到多個feature selection，在之後的研究領域中可能激發我去更換不同的features以達到更精確的model這樣的操作練習。

1. 參考資料