Introduction-to-Information-Retrieval-and-Text-Mining, HW 3

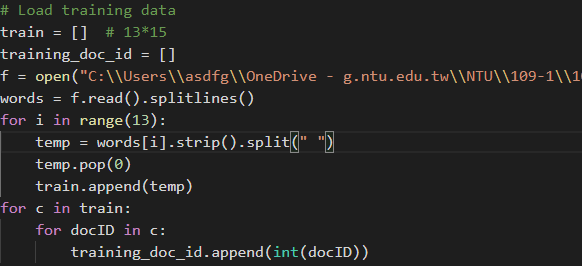
B07302230, 財金三 王博奕

1. 執行環境：Visual Studio Code
2. 程式語言：Python 3.8
3. 執行方式：
   1. Pip install nltk以便於接下來使用Porter’s algorithm 及 stopword。
   2. Pip install math 以進行log運算。
   3. Pip install re 以便切字。
4. 作業處理邏輯說明：

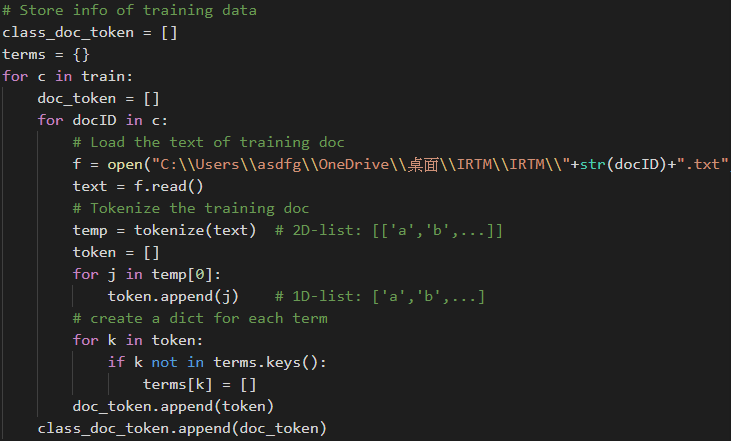
首先，tokenize的函數沿用作業二的部分，因此不再贅述。接下來程式主要分成四大部分進行，以下將分點描述。最後再提出改進的地方。

1. **處理Training data:**

載入作業提供的訓練資料，存為13\*15的二維list(train)，第一層為class、第二層為該class底下的文件。此處我另外建一個training\_doc\_id的list，專門存訓練資料的號碼，以便後面排除。

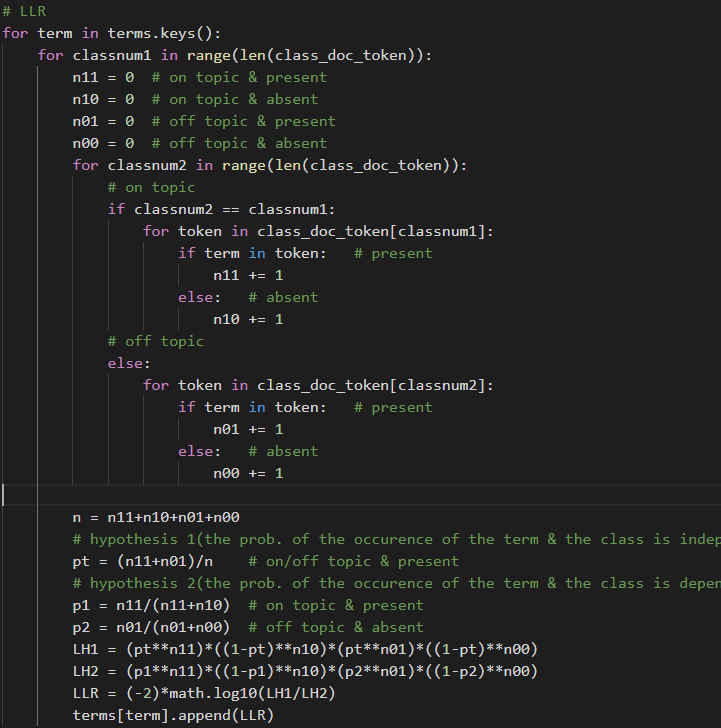


接下來我建一個三維的list(class\_doc\_token)，第一層為class、第二層為class底下的文件、第三層為各文件的token。這裡我先用temp來執行tokenize，再轉為token是因為當初tokenize寫法的關係，原先的的temp是二維的，經降維成token。我在這裡也先建terms這字典，是先存訓練資料的token，屆時這個字典的value將儲存LLR算出的各個分數。



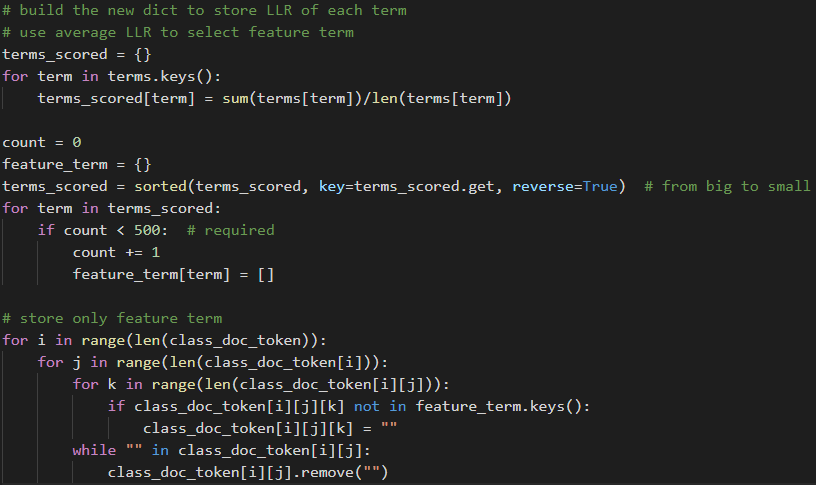
1. **Log Likelihood Ratios:**

這次作業我使用的Feature Selection的方法為LLR，計算方法即為上課ppt所記載的。這裡的目標為選取LLR分數高的term。而terms這字典的value為各個class其LLR的分數(一個有13個elements的一維list)。

****

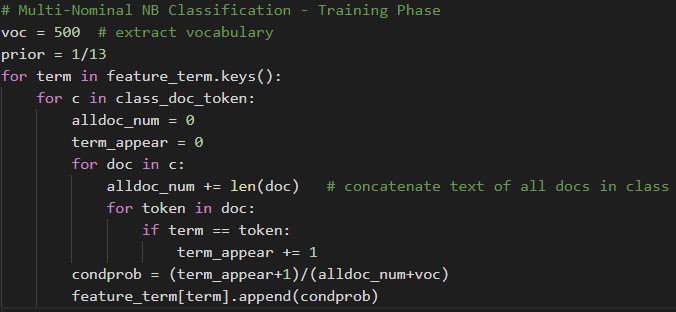
1. **篩選500個最有力的feature term:**

原先的terms這個dict的value是個list，包含了這個term under different classes時不同的分數，而我這裡直接把list裡的element直接取平均算出一個平均LLR，最後挑出前500個高分的feature term，作為篩選term，這時的feature\_term這個dict只剩500個key。有了feature\_term，接著就把原先的class\_doc\_token這個list中的token部分只留下feature term。



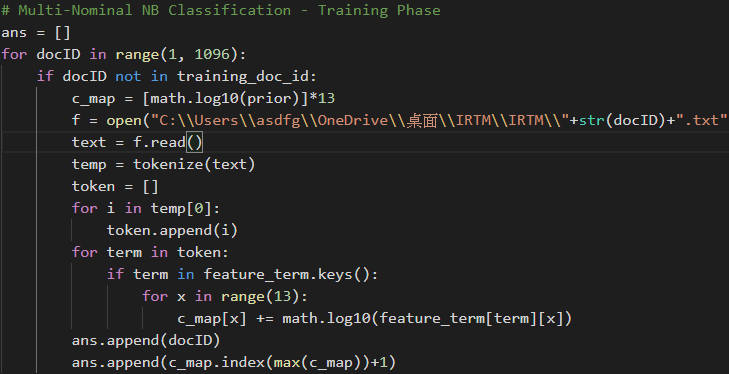
1. **Multi-Nominal Naïve Bayes Classification:**
   1. **Training phase:**

這裡就將利用老師上課ppt所提供的Pseudocode來完成，值得一提的事是因為training data提供的樣式關係，可以直接知道prior是多少。另外，我也將各個詞的條件機率加到feature\_term這個字典的value。



* 1. **Testing phase:**

最後將每一篇的文章先tokenize後，再來針對feature term來計分，最後選大的來當作該篇為哪個class的依據。



1. **Improvement:**

結束了以上程式碼，我在最後再試圖調整了模型，其中我發現收太多feature term會導致分數降低(可能原因是有雜訊)，因此我把feature term選取數量降低為一半，也就是250個，發現在kaggle上的成績比較好。