HW3 Documentation 資管碩一 r06725015 李尚恩

1. 執行環境：Windows，Jupyter
2. 程式語言：Python 3.6.2，需要 nltk、pandas 與 numpy套件
3. 執行方式：（IRTM 與 training.txt 需在同個資料夾）

先使用 pip 安裝 nltk、pandas 與 numpy

在 command line 上輸入：

python HW3.py 或 python3 HW3.py 或 python3.6 HW3.py

執行完畢後，可在同個資料夾底下看到 output.txt。

1. 作業處理邏輯說明：

以下邏輯處理以程式段落順序說明：

1. 第一段：import 所需要使用的 python 套件，並從 nltk 中取得 stop words 的清單。定義一些全域變數，包含類別數與文章路徑。
2. 第二段：讀入 training.txt 取得 training 要用的文章編號與對應的類別，並切割成 training 與 validation set，不過這部分並沒有使用到，最後的結果並沒有切 validation，因此 valid\_size = 0。
3. 第三段：定義 feature selection 所需的 fuction，以下逐一說明：
   1. ExtractVocabulary(doc)：輸入一個文章編號的清單，會輸出所有文章中出現的字的字典，其中經過stemming與stopwords的處理。
   2. BuildFeaturesLabels(doc\_class, vocab)：輸入文章編號與對應類別的清單與剛剛建立好的字典，會輸出features與labels，前者為一個陣列中存著每篇文章的bag of words，而後者則是存著每篇文章對應的類別。此儲存方式是便於後續篩選的運算。
   3. ComputeChi(features, labels)：實作Chi-square test，因為作業是一個Multiple Classifier的問題，所以我採取的是先分別計算每個class對每個t的A(t,c)，最後再取平均。
   4. SelectFeatures(all\_vocab, features, labels, num)：根據ComputeChi的結果取出前num高的term編號。
4. 第四段：定義 NB classifier 所需的 function，以下逐一說明：
   1. ConcatTextInClass(doc\_class, c, vocab)：將類別c的文章中的字全部append到一個清單中，其中會判斷該字是不是屬於剛剛建立好的500字清單。
   2. TrainMultinomialNB(train\_doc, vocab)：實作NB演算法，依照老師投影片上的方式，先計算P(C)再透過ConcatTextInClass計算條件機率的部分，最後輸出訓練完成的機率模型。
   3. ExtractTokensFromDocs(vocab, d)：取出文章d的tokens，且該token必須是在vocab裡面。
   4. ApplyMultinomialNB(vocab, prior, condprob, d)：先透過ExtractTokensFromDocs來取得文章中的tokens，再使用訓練完的機率模型來計算該篇文章對各類別的分數，最後取最高分的作為預測的類別。
5. 第五段：進行 feature selection 挑出最有預測能力的 500 個字。
6. 第六段：使用上一段做出來的字來訓練NB的機率模型。
7. 第七段：將尚未有標記類別的文章取出，並逐一預測其類別結果，最後將預測完的結果存入 output.txt。
8. 心得：

這次作業主要比較花時間的地方也是在儲存資料上，要如何有效辨別哪些資料是需要儲存的，哪些資料是要用到再去拿就好，我覺得要先思考好再開始寫，比較不會很混亂。另外，在function的設計也是一樣的問題，有很多類似的功能其實應該可以用一個function就夠，但是因為一開始沒有思考太多就寫了，導致function之間有點重複。經過這樣的實作更加了解了NB演算法的運作，也感受到其預測的能力。在實作feature selection的部分也是很有感覺，我有實際印出被選出來的字，的確都是一些比較特別的字，不會是每篇文章都會出現的字，例如：Japan、Monday。雖然自己實作的成效可能不比現成的套件，但是有了這些經驗後，在使用套件時就可以更加上手。