

1. 執行環境

Jupyter Notebook

2. 程式語言 (請標明版本)

Python 3.7

3. 執行方式 (重要!!!!!!!)

以 Jupyter Notebook 執行

4. 作業處理邏輯說明

環境：

- import 套件，以及設定好檔案路徑
- pprint 只是讓 output 好看一些，不全部擠在一起

先把老師上課 ppt 中的程式架構列出，主要包含 Training Phase 跟 Testing Phase
照著裡面需要的功能一一列出

Training Phase

1. ExtractVocab

- ExtractVocab 的部分可以延續作業二，並且有回去看當時的檔案 output，盡可能把一些標點符號、一個 letter 的刪掉
- 另外本來還想把 url 刪掉，後來 feature select 的時候也會篩掉因此不另處理
- 上一次作業有個疏失，先 stem 了才 remove stopwords，導致部分 stopwords 沒有清乾淨，這次將順序對調，先 remove stopwords，最後才 stem
- 這次就不用 counter 函數了，因為老師給的程式架構當中要把每個 class 的 text 整合一起，因此先做 token 而已

2. CountDoc

- 這部分把作業 PA-3 中的 <https://ceiba.ntu.edu.tw/course/88ca22/content/training.txt> 網址擷取 class_id 跟 doc_id，還有各類別文件數、總數
- 最後因為要區分 testing data，所以也做了所有 training data 的 doc_id 清單

3. TEXTc <- ConcatenateTextOfAllDocsInClass(D, c)

- 按照每個類別的訓練文件，各自整合在一起
 - 這邊原本要用上次 collection 套件的 counter 方法，計算 tf, df，但是後來聽助教說用 set() 這個方法，感覺比較方便
- 結果：

```
In [50]: 1 pp.pprint(count_tf)

{ 1: { 'abl': 2,
      'aboard': 4,
      'accid': 9,
      'accord': 2,
      'acknowledg': 1,
      'actual': 2,
      'admir': 1,
      'advanc': 1,
      'affair': 1,
      'afternoon': 1,
      'agenc': 1,
      'aggress': 1,
      'ago': 1,

In [51]: 1 pp.pprint(count_df)

{ 1: { 'abl': 2,
      'aboard': 4,
      'accid': 5,
      'accord': 1,
      'acknowledg': 1,
      'actual': 2,
      'admir': 1,
      'advanc': 1,
      'affair': 1,
      'afternoon': 1,
      'agenc': 1,
      'aggress': 1,
      'ago': 1,
      'aim': 1,
      'akishio': 1,
      'alex': 1,

In [53]: 1 print(len(Text_of_All_Docs))

5436
```

4. Feature Selection, 採用 LLR(助教的建議, 而且文獻指出 LLR 效果比 Chi square 好)

因為 Vocabulary 數量太過龐大, 會有太多 noise, 所以用 feature selection 挑出代表字, 一樣先附上老師給的程式架構做為參考, 並參考 PPT 的說明, 得知需要的變數有 n11, n10, n01, n00, Pt, P1, P2, H1, H2

- 因為要從所有 training 文件中取得代表字, 因此需要 Text_of_All_Docs 作為參數
- 計算 n11, n10, n01, n00 需要是有出現在 training doc 中的 term 的 df, 因此需要 count_df[class_id][term]中的[df]
- 計算出每個 term 的 LLR 分數
- 把每個 class 都取出最高的前 number_of_feature 個代表字, 取 38 的話全部代表字為 498, 最為接近, 39 的話則是 510。
- total_feature_maxnum 所有代表字加總最多 500
- train_class_doc_num 因為需要知道每個分類有多少訓練文件, 以計算 n10, n00

5. 計算 Tctk, Tct', 以及每個 class 的 Prior, condprob

- 此例每個 class 都是 15 個訓練文件，因此 Prior 剛好都一樣
- `condprob` 每個代表字出現在該 class 的機率
- `Tct` = The total number of terms in D from class c.
- `Tctk` = The number of occurrences of tk in D from class c
- 代表字若有出現在該 class，則給他該代表字的 `tf` 值；若無，幫他+1 做 smoothing，`Tct` 則是所有出現在該 class 的代表字的 `tf` 值加總
- `Tctk` 則是指該代表字出現在該 class 的 `tf` 值

Testing Phase

- IRTM 中所有文件，但排除訓練文件，為測試資料
- 取出字做法如同 Training Phase
- 每個 class 的 Prior 分數取 log
- 計算該文件為各個 class 的可能性
 - 若該字在代表字中，將他該 class 的分數加上 `condprob[class][term]` 的分數，持續加總到跑完該文章所有字

結果：

```
This is Doc 17's score
{ 1: -74.26721950708146,
  2: -50.18002788418403,
  3: -74.61944496368773,
  4: -68.57587600777788,
  5: -74.03978118892391,
  6: -73.20169484318662,
  7: -74.88487922893816,
  8: -70.9657828023104,
  9: -72.70129267369653,
  10: -66.21104489848729,
  11: -72.10836895486187,
  12: -71.0961747242943,
  13: -71.61665608818785}
This is Doc 17, it should be class 2
This is Doc 18's score
```

Write Result

轉為 `dataframe`，再用 `to_csv` 的方法寫成 `result`

心得：

每天寫一點程式都像在跑馬拉松，但是多謝于真助教那天的開示，知道方法步驟後可以開始的很快！但是 `prediction` 的結果我真的大吃一驚！！其實比較想聽同學們其他 `feature selection` 的分享．．．