|  |
| --- |
| 運用工具 & 資料前處理說明 |
| 1. 透過nltk的word.tokenize function，將Documents與Queries進行斷詞。 2. 運用snowball\_stemmer function將詞性還原。 3. 使用nlt.corpus中stopwords篩選掉不帶有資訊的字詞，ex: the, a, and…。 |
| 第一次檢索：rocchio\_bm25.py、bm25\_main.py |
|  |
| 1. Documents TF計算：   計算該文章每個字出現的次數，並運用Log Normalization ( 1+log(tf) )概念平滑TF參數，以完成該doc各文字的TF計算。   1. Query TF 計算：   計算query各文字出現的次數則是運用最原始的TF計算方式。   1. IDF計算:   為最原始的IDF計算方式：log((N-ni+0.5)/(0.5+ni)) |
|  |
| 第一次檢索為BM25L結合BM11，經由實驗參數設定結果如下：  K1 = 0.8、b = 0.7、K2 = 0.07、K3 = 0.2、delta = 0.2，  此外，由於「平均長度-長度」有可能為負值導致所求分數減少，因此我試套上絕對值後準確度的確會提升。再者，在實作 𝑡𝑓𝑖’j時前面有多X 0.8，該參數亦是經由實驗而進行設定，而我認為該參數設定是要將Doc Term Frequency的值更平滑。 |
| 第2次檢索：rocchio\_vsm.py |
| 使用上述bm25之模型找出的每個query前10篇相關的relevant doc，將上述所得之結果套用至第2次檢索，而我所使用的第2次檢索之模型為vsm，套用的tf平滑方法同上，idf為log(N/(ni+1))。經由實驗結果，拿前10篇相關的relevant doc，並以參數α=1，β=0.2為最好的實驗結果。 |
| 心得 |
| 其實從上次作業開始，難度就爆炸性上升，感覺自從開始教PLSA後，就有點跟不上進度了...，而且感覺後面幾次的作業關聯程度都很高，故由於上次作業的EM就已經沒做出來了，我自己後來也有再嘗試研究並理解，但因概念上就有點抽象了，還需將其轉換程式碼真的很困難，故這次作業我一樣沒有用到那些較進階的model (ex: SMM、RM…etc)，只運用Rocchio方法。此外，由於我第一次檢索是用bm25，第2次檢索是用vsm，我也曾嘗試過將第2次檢索改成bm25，但得出的準確率非常低，而因為準確率過低的關係，可得知我第2次bm25之檢索方式一定有寫錯之地方，雖然我目前還是不知道錯在哪裡，但慶幸的是有把VSM的做法做出來。  當然，從看大家連過Baseline都很難開始，外加幾位認識的資工同學也是很苦惱，甚至老師還把繳交作業的時間延長。就可以看出這次作業難度確實很高。但同時也看的出來有很多人在後來延長的第3周有成功做出來，大家的分數都突然變得好高! 雖然這時看到其實也是蠻挫折的，因為感覺有很多人在最後有做出來進階型的model，又或是他們都有把Rocchio做得更準確，但我自從第一周做完Rocchio就一直停滯了，所以我很希望這次的作業老師可以找更多人上台分享他們怎麼做，希望藉由同學上台的分享可以讓我更了解。 |