|  |
| --- |
| 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…。 |
| 心得 |
| 由於這次作業的難度真的是爆炸性上升，光是聽懂老師上課在說什麼就很吃力了。連一開始的kmeans分類我就遇到很大的困難了，自己也嘗試做過於kmeans\_collection.py，雖然最後其實沒有用到…，但我有嘗試去做且透過此次作業稍微了解kmeans。而於EM的部分，一開始真的很茫然，到處問資工同學甚至連師大的學生都被我叨擾了，但大家起初也不太理解怎麼做，不過還是很感謝他們的幫助。當然，我有上網自己研究也嘗試自己修改，但由於資料量太大，若使用三維的結構會導致memoryerror，無論我怎麼修改或上網搜尋都還是會報錯，故這份作業我沒有用到EM algo，只用最基本P(w|D)P(w|BG)，以下會述說參數的比例及作法。但到今天我才發現原來有人在Kaggle上說使用稀疏矩陣就可解決memoryerror…，故我會再找時間研究並且嘗試做出來。 |
| P(w|D) P(w|BG) 參數設定- background.py |
|  |
| collection\_docname.txt：作法為「每篇文章的字/該文章的長度」，即是P(w|D) |
|  |
| BGLM.txt：作法為「該字出現在所有文章次數/corpus length」，即P(w|BG)，而本模型之corpus length為unique，經過測試該作法所獲之正確率較高。 |
| 相似度計算-PLSA\_final.py |
| 本作業使用方法為alpha\* P(w|D)+beta\* P(w|BG)，其中alpha經過測試設為0.988，beta為0.012，由這兩參數所求知準確率會較高，而 P(w|D)和 P(w|BG)參數由上述預處理的 collection\_docname、BGLM.txt所獲得。 |