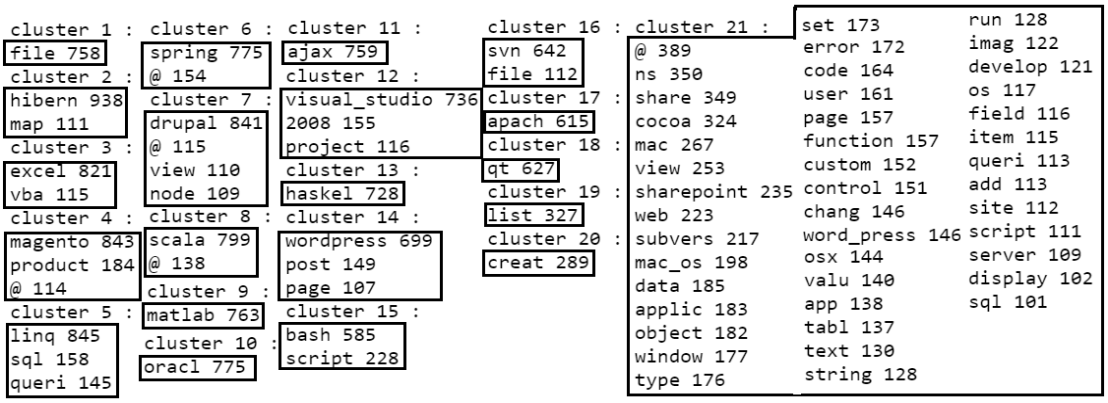
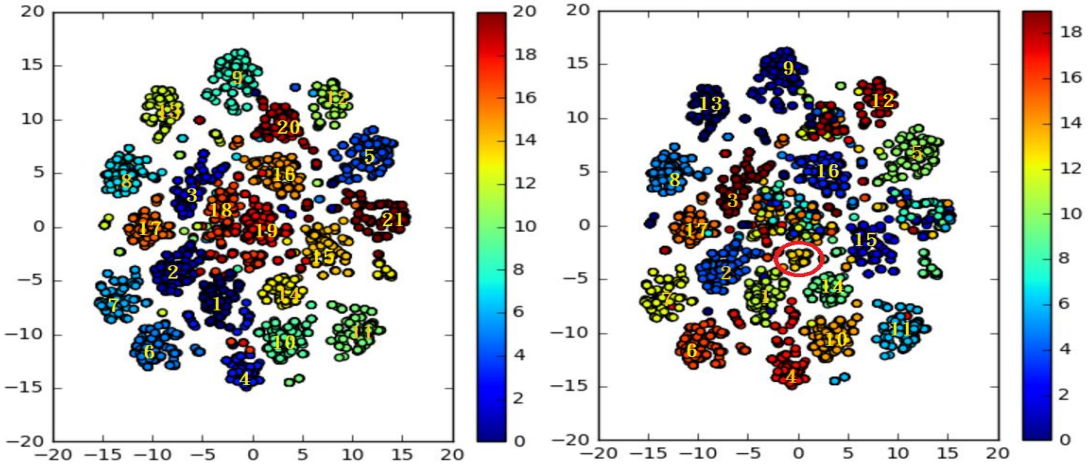
1.對每篇文章的 title 依序做：camel case、轉小寫、移除標點符號和 stop words(這裡使用 nltk 內建的和網路抓的 stop words 共四百多個字)、詞幹提取(stem)、word to phrase 等後，再用這些 titles 建立 BoW(只抽取出現頻率最高的 500 個字當作表示每個 title 的維度大小)在拿去做 k-menas 分群(分成 21 群，前 20 群以字頻最高的 20 字的維度當作 centroid，例，cluster 1:[1,0,…,0],cluster 20:[0,…,0,1]；第 21 群的 centroid 設為原點)。則在每群中最常出現的字如圖一所示：

圖一、在每群中出現頻率超過 100 次的字(與其出現次數)，'@'表數字 1~9



此種分類方式的 F-Measure 約為 86%(相較之下隨機從字頻最高的 500 字中挑選 20 字的維度當作 centroid 的 F-Measure 約為 58%)，第 21 群的用意為分出較無法分在前 20 群的 data(在倆倆比較 data 是否為同類時若至少有一方屬於此類、則一律視兩者為不同類)，由圖一可知此群的常見字比其他 20 群多更多(前 20 群因為高頻字較明顯而集中所以用 BoW 效果會較好，或許可以說這些高頻字可能可以代表此群)，因此此群中的 data 確實較不易在 BoW 的方式下辨別出其屬於哪群。在經計算後得出 TF-IDF 值最低(約為 0.1)的字為'file'，即使在移除一般文章中常見的 stop words 後仍可得到 TF-IDF 值如此低的字可能是因為此字屬於特定領域(電腦科學)中的常見字(且由圖一可知'file'同時常見於第 1、16 群)，若移除此字後再以同樣方式進行分類可以得到 F-Measure 約為 89%的結果，可知'file'雖不為一般文章中常見的字，但在電腦科學領域類的文章中卻是常見且略顯多餘的字。

圖二、左圖為我做的分類，右圖為 true label，黃色數字表每群代號(第幾群)



2.用第一題的方法 (未移除'file')建立 BoW 得到代表每個 title 的 features 後，先用 PCA 將 500 維的 features 降為 50 維後，再每群抽樣出 250 個 data(因為記憶體

不足的問題所以未使用所有 data)拿去做 tSNE 降為 2 維(最後代表每個 title 的 vector 長度為 2)可畫出圖二左圖，右圖為同樣的(抽樣)data 換上真正的 label。

在左圖中，可以發現每群分的還蠻清楚，第 15 群分的較散的原因很可能是其最常出現的字'bash'出現的頻率較低只有 585(其他 19 群最常出現的字的頻率大部分都超過 600)而第二常出現的字'script'的頻率卻高達 228(其他 19 群第二常出現的字的頻率都小於 200)，且'bash'為第 1 群中第二常出現的字、'script'在第 21 群中出現次數也高達 111('bash'和'script'的 TF-IDF 在所有 title 中最大只有 0.1 多，他們不具有高鑑別力)，所以與其他群相較之下(尤其是那些最常與次常出現字數相差較大且這些字不會是其他群中的高頻字，例如第 2、8、10、12、13、17 群)data 間的距離較不緊密。另外第 19 群不只分的散而且幾乎快要和第 18 群攪和在一起，該群中最高頻率字'list'出現次數只有 329 且 TF-IDF 值約 0.14，因此第 19 群以'list'的維度作為 centroid 可能不恰當(第 20、21 群也有類似問題，前者的 data 甚至散布到第 14、15 群間，若劃出所有的 data 而非取樣應該更明顯)。第 21 群專收無法分類在前 20 群的 data，其高頻字的出現次數相對前 20 群較為平均且可能含括其他群的高頻字，如第 14 群有部分 data 會顯示在第 21 群下方可能是因為'page'為他們共同的高頻字、第 5 群與第 21 群位置相近可能是因為'sql'與'queri'為他們共同的高頻字、第 15 群與第 21 群位置相近也可能是因為'bash'和'script'為他們共同的高頻字。

右圖中可發現第 1~17 群(紅圈圈出的第 18 群相對有點小，即第 18 群的分類效果相對較弱)分類效果依然不錯(用第一題的方式做分類所得到的準確率約可達 17(群)/20(群)=0.85，第一題已提及此種分類方式的 F-Measure 約為 86%)，而剩下 3~4 群(此種分類方式的第 21 群本來就是專放那些難以分類的 data)卻攪和在一起了，或許在此種分類方式下最多只能分類出約 17 群的原因為：(我做的)第 19、20 群中並沒有高鑑別力的字可以用來分群，所以這可能就是使用 BoW 的限制。

3.以下主要列出六種分類方法之比較(下列所有方法所使用的 title 已做過 camel case、轉小寫、移除標點符號和 stop words、詞幹提取、word to phrase 等處理)：

a.TF-IDF -> K-means:將每個 title 的 word 以 TF-ID 值表示所形成的(用來表示每個 title 的)vector 拿去做 K-means 分類(分 20 群)後得到的 F-Measure 約為 18%；

b.加入相似字 -> TF-IDF -> K-means:先以 gensim 內建的 word2vec(每 50 個 iteration 令 learning rate(初始:0.025)減少 0.002 共做 10 回)將所有 title 的字彙 train 成可以用 vector(設為 128 維)表示的形式，再利用這些 word vector 找出與其最相近的前 3 個 words 加入每個 title，再將已加入相似字之 title 的 word 以 TF-ID 值表示其 vector 並拿去做 K-means 分類(分 20 群)後得到的 F-Measure 約為 24~28%；

c. word2vec -> K-means:用 word2vec 將所有 title 與 document 的字彙 train(約 500 iterations)成可以用 vector(128 維)表示的形式後，將出現在 title 的 word 其 vector 相加取平均後拿去做 K-means 分類(分 20 群)後得到的 F-Measure 約為 39~57%(iterations 越小、少做任一種(或一種以上)對 title 的處理之分數會越低)；

d. word2vec -> K-means -> 比對核心字+ K-means label: 用 word2vec 將所有 title 與

第 3 題中，因部分方法有與 r04922081 江宗臻同學討論過，故或許相似。

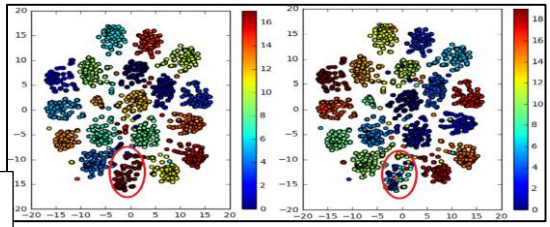
document 的字彙 train 成可以用 vector(128 維)表示的形式後，將出現在 title 的 word 其 vector 相加取平均後拿去做 K-means 分類(分 20 群)，用(20 個) K-means centroids' vectors 去找出最接近的字(cosine similarity 大於 0.7 就算是，共找出 16 個，這些字可能對每一群而言具有一定的代表性故取之)，若發現兩個 titles 間有一個以上的字相同且該字是這 16 字中的任一個，即算同類，否則還要再比對 K-means label 是否相同，此種分類方式的 F-Measure 約為 52%(在相同條件(當時少做一些對 title 的前處理)的 c 方法只有 39%，可以推估"比對核心字"有效)；

e. 加入相似字 -> 比對相同字數: 先以 gensim 內建的 word2vec 將所有 title 與 document 的字彙 train 成可以用 vector(設為 128 維)表示的形式，再利用這些 word vector 針對那些 word 之字頻介於 α 與 β 間，將與其之 cosine similarity> γ 的 words 加入 title，若兩個 titles 間有超過 ε 個 word 相同則視為同類，此種方法根據 α, β, γ, ε 不同的大小設定之 F-Measure 約為 68~78%(最佳 α=3200, β=1700, γ=0.41, ε=7)；

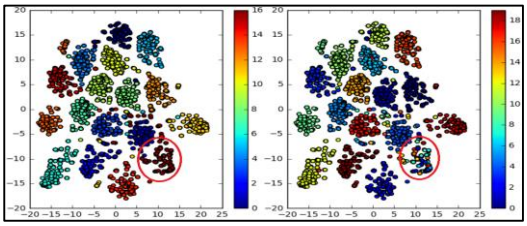
f.BoW(取 500 維) -> K-means:即第一題所述之方法，F-Measure 約為 88~91%(分數根據不同分群與設定 centroid 方式有所差異，詳見第 4 題)。

4.根據第一題所述之分類方法(用 BoW 取 500 維做 K-means 並濾掉 TF-IDF 值較低的字如'file')，若只分類 20 群則 F-Measure 約為 17%，但是若分成 20+1 群(如第一題所述需要自己先設定 centroid:前 20 高頻之 word 的維度當 vector+原點) 則 F-Measure 約為 89.3%，若分成 21+1 群(如第一題所示需要自己先設定 centroid:前 21 高頻之 word 的維度當 vector+原點) 則 F-Measure 約為 88.5%，分成 19+1 群、18+1 群、17+1 群、16+1 群之 F-Measure 依序約為 89.9%、90.5%、91.0%、2%，根據實驗結果分類 17+1 群之效果最好(嚴格說起來' 17+1 群'應為'18'群，雖然最後一群的用意是專收較無法分類的 data—若兩筆 data 有任一方屬此群則一律視這兩筆 data 不同類)。如下圖，17+1 群的前 17 群確實能有效分類，而最後一群呈現在 truel label 的分類上卻是難以分類(同一群卻有多種顏色)；16+1 群除了少掉一群能有效分類的群外其他情況也與 17+1 群差不多，但兩者 F-Measure 卻差這麼多也許是因為程式 bug，因為即使 16+1 群只能有效分 16 群，只要在 16 群內的 data 數足夠多且 test data 沒有偏頗，F-Measure 應約為 16(群)/20(群)=0.8；

17+1 群

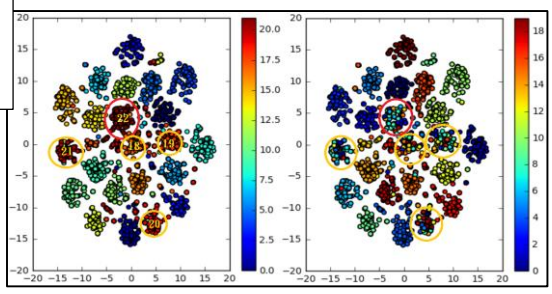


16+1 群



左圖:我的分類
右圖:true label
紅圈:最後一群

21+1 群



而 21+1 群的有效分類群數為 17 群，然而因為比 20+1 群(圖二)多一群在 truel label 的分類上會使 data 難以分類，所以 F-Measure 會較後者低。

由實驗得知，因為在 BoW+K-means 分類方法上不論分幾群，在 truel label 分類上能明顯有效的分類群數最多為 17 群，所以當分成 17+1 群(18 群)時效果較好。