Q1. 高頻字詞分析和移除不相干字詞

把 `title_StackOverflow.txt` 去符號、轉小寫後，分別用 BoW 和 TF-IDF (取 IDF 部份) 分析字頻分佈。下表列出兩方法找的前 20 高頻字、平均值、標準差。一些不重要字詞和 stop words 會很明顯分佈在 3 個標準差之外 (表格灰底部份)。雖然 IDF 把多次出現在同一文件的字只算一次，但兩者排序大致相同，影響不大。(常見字如 'a' 或 'i' 沒出現是套件 `TfidfVectorizer` 和 `CountVectorizer` 的原因，詳情看此連結)

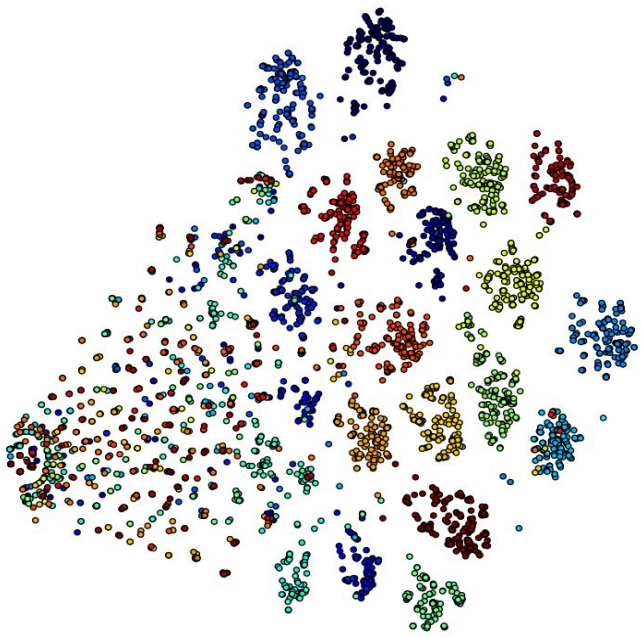
BoW 最高單字			
to	6574	from	1420
in	5942	do	1294
how	4017	using	1168
the	2841	an	1151
of	2084	can	1062
with	2034	hibernate	935
and	1854	what	894
for	1829	excel	888
is	1645	wordpress	882
on	1567	magento	879
平均 227.812，標準差 513.142			

IDF 最低單字			
to	2.238	from	3.656
in	2.269	do	3.760
how	2.608	using	3.844
the	3.078	an	3.908
with	3.308	can	3.938
of	3.316	hibernate	4.082
and	3.414	what	4.115
for	3.421	wordpress	4.123
is	3.530	magento	4.134
on	3.577	excel	4.150
平均 6.103，標準差 0.888			

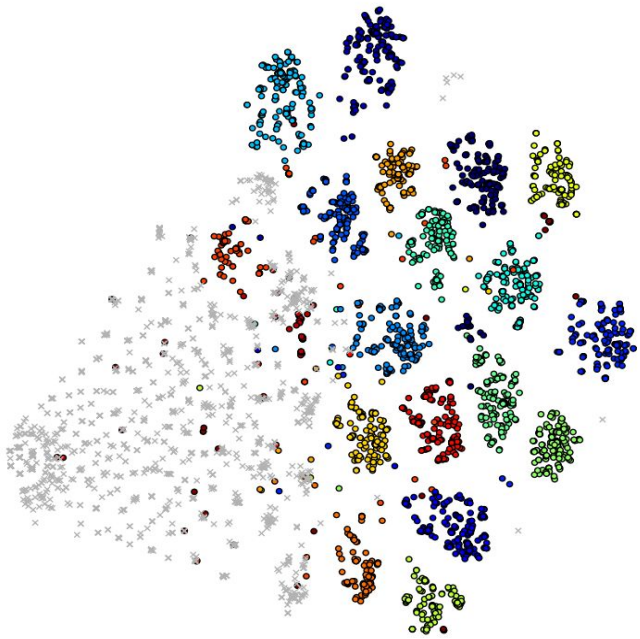
知道 tag 的情況下可濾掉 BoW > 1000 或 IDF < 4.0 的字。但我們不能知道 tag，只可猜測 tag 也很高頻，故過濾時得保守一點 (如取 3 個標準差外)，因此效果有限。實際上用現有 stop words 字庫會是更好的選擇！

Q2. 視覺化分群: 解答 vs. 我的分群

用 BoW 當 feature (500 維)，先用 PCA 降到 100 維，取 5000 筆標題 (各分類數目一樣)，再用 t-SNE (`perplexity=60`) 投影平面作圖。**圖 1** **圖 2** 分別是用正解 label 和我分群的標色結果。



1

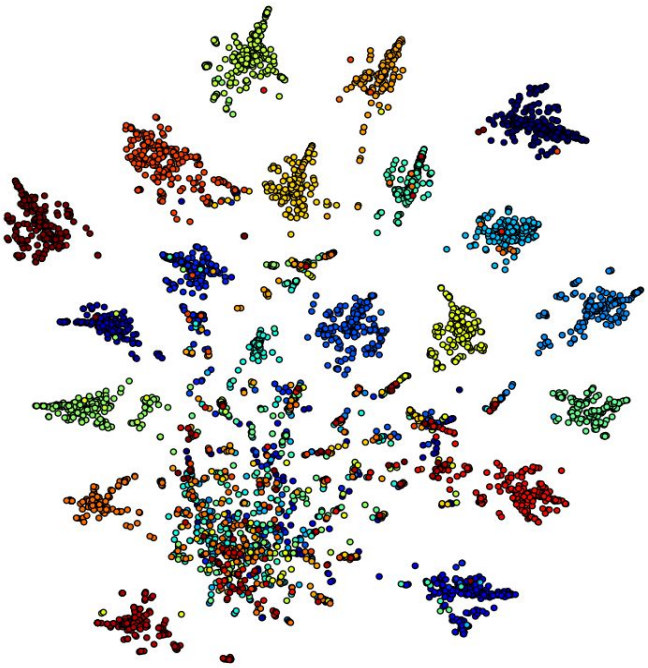


2

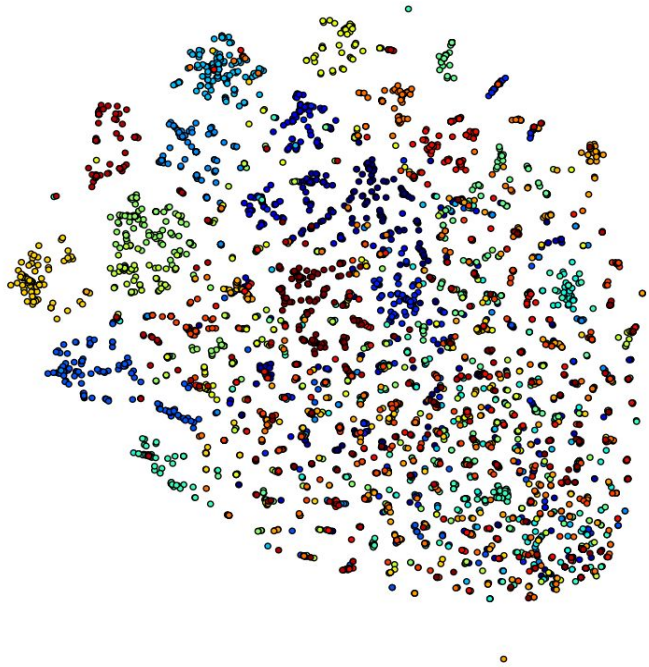
不同色代表不同群，從左圖發現 tag cocoa (中偏左的淡藍色) 的點特別分散，對應標題觀察，發現很多單字都和 cocoa 不相干，甚至有些其它分類的關鍵字 (如 mac、osx)。

不必知道解答，從分佈看就發現有超大一群散亂資料！把這群另歸為 “其它” 類 (右圖淺灰 x 標籤)，任兩點包含至少一個 x 則判斷為不同類 (**Q3** 方法 **e**)，會大大提昇效果！右圖是用我的方法分 21 類，即 20 類 tag 加一類 “其它”。實際上每次 t-SNE 後 “其它” 分類的點位置分佈都不同，例如有時散在右邊，我取剛好分佈在左下的圖，方便觀察。

圖 3 以平均 word vectors 當 feature (**Q3** 方法 **c**)；**圖 4** 以 TF-IDF 當 feature (**Q3** 方法 **a**)。和我實驗結果呼應：TF-IDF 在本次作業效果並不好。



3



4

Q3. 不同實驗設定

a. TF-IDF + *k*-means

前處理在不同實驗設定都一樣：首先用 NLTK 內建和網路找的共兩個 stop word lists 移除不重要字詞。接著全轉小寫、移除標點符號。再把可能是 phrase 的字綁一起 (如 visual studio 轉 visual_studio)，使用套件為 Google 的 `word2vec.word2phrase`。

接著用 TF-IDF 當特徵 (維度 500)，再用 *k*-means 分 18 ~ 25 群，F-measure 大約只有 20% ~ 30%。而且同樣參數不同次 *k*-means 仍有不小差異 (最多 5%)。Feature 分佈如**圖 4**。

b. Word Vectors 擴增相似字 + TF-IDF + *k*-means

前處理同上。接著用標題訓練 word vectors，然後用 word vectors 對標題單字找相似字加入 (類似 query expansion)。我推測關鍵字 (如 tag 本身) 應很高頻，故只針對高頻單字 (如前 30%) 做相似字擴充。擴充方法嘗試過加入最相似 3 ~ 6 個字、取 cosine similarity 超過 0.6 等各種排列組合，但大同小異。最後同前方法用 TF-IDF 和 *k*-means，分數約 33% ~ 38%。Feature 分佈類似**圖 4** 但好一些。

c. 平均 Word Vectors + *k*-means

前處理同上。一樣用 word vectors 擴充高頻相似字。但改成把標題所有字的 word vectors 取平均當該標題 feature，再用 *k*-means 分 20 群，分數約 40% ~ 50%。Feature 分佈如**圖 3**。

d. Word Vectors 擴增相似字 + 直接比同字數目

前處理一樣，也用 word vectors 加入相似字。但這次不分群，而是在比較兩標題是否同類時，直接看兩標題有幾個同樣的字。至於幾個字相同就算同類則是靠 trial and error，而且這隨著加入相似字方法

而不同，效果也很敏感。和黃郁庭同學討論和測試 (i.e., 浪費 *N* 次 Kaggle 額度 Q_Q) 發現從 40% ~ 80% 都有。

試到最好的設定是：對字頻 > 100 者找 6 個最相似字，若 cosine similarity > 0.4 才加進來，兩標題有超過 7 個字相同則判斷同類，分數有 80%。但這方法實在太莫名其妙根本暴力亂試而已 Orz

e. BoW + *k*-means (多加 “其它” 分類)

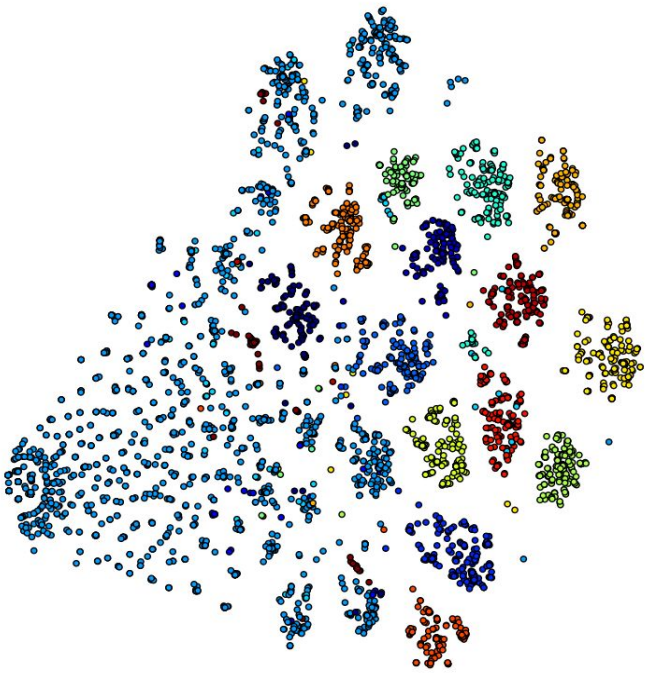
此方法簡單但秒殺前面我的亂搞方法，感謝高滿馨同學教我 m(_ _)m 前處理都一樣，然後用 BoW 當 feature。重點在觀察 visualization 分佈 (**圖 1**、**2**)，會發現除了將近 20 個的群聚，還有一大群散亂的點，應該是該標題內毫無相關的字詞，將這群分類到 “其它”。先想辦法用 *k*-means 分 21 群，再找到 “其它” 那群。判斷兩標題是否同類時，只要其中一方是 “其它” 就當成不同 (輸出 0)；若兩者都不是 “其它” 則看是否被分到同一群。我試過兩種分成 21 群的方法：

- 找 20 個 feature 彼此互相垂直的標題當 *k*-means 初始中心，再多加原點 (即 `[0, 0, …, 0]`) 當第 21 個中心，用 *k*-means 分 21 群。以原點為中心那群則是 “其它” 類。
- 先用 *k*-means 分 25 群，把數目最大的 21 群中心抓出再分群，其中數目最大者是 “其它”。

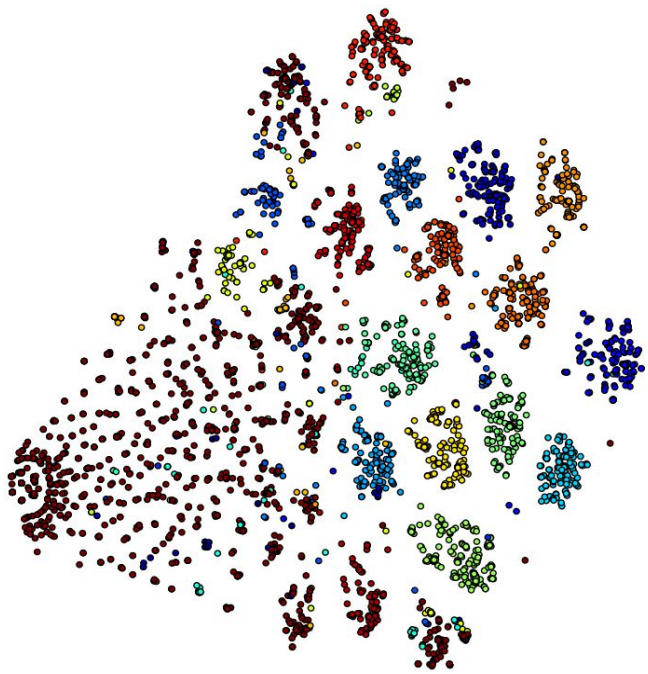
兩者方法 F-measure 差不多都有 87%，而且可以再高！用上述方法 train 好幾個 *k*-means，判斷兩標題是否同類時，讓這些 *k*-means 投票表決 (如判斷 0 者多則輸出 0)，準度可達 91%。

Q4. 比較不同分群數

圖 5 分成 17 群；**圖 6** 分 23 群。分得少的好處是較能把不知道不明瞭不想要分類的點分到 “其它”；分得多好處是有些聚集程度小的分類，仍有部份可分到一類。



5



6

以我效果最好的方法 (**Q3** 方法 **e**) 來做實驗，初始化 *k*-means 中心時，群數對 “其它” 類的初始中心影響不大。但希望找到剩下 20 群中心，所以先分多一點 (如 25 群) 再挑群數最大的 20 個中心，比較不會遺漏。最後包括 ”其它” 類，分成 19 ~ 22 類效果最好 (F-measure 平均 90%)。

但分群數目似乎依實驗方法而有不同，和所取 feature 分佈有關。例如我用 **Q3** 方法 **a** 和 **b** 要分很少群效果較好，**圖 4** 分佈可解釋之；**Q3** 方法 **c** 則是 20 群最好 (**圖 3**)。