

1. Analyze the most common words in the clusters. Use TF-IDF to remove irrelevant words such as “the”.

沒有加入 **stopword** 時，最常出現的字是：wordpress, excel, matlab, visual, bash, spring, hibernate, ajax, drupal, linq, haskell, magento, **to, and, on, for, do, is, of, with** 可以發現除了正確的 **label** 之外，仍然出現了很多無關緊要的字，可能是因為這些字其實在文件中出現的機率並非如此頻繁，因此可能誤被當作重要的 **feature**。

在加入 **stopword** 之後，最常出現的字是：wordpress, oracle, svn, apache, excel, matlab, visual, bash, spring, hibernate, scala, sharepoint, ajax, qt, drupal, linq, haskell, magento, mac, **file**

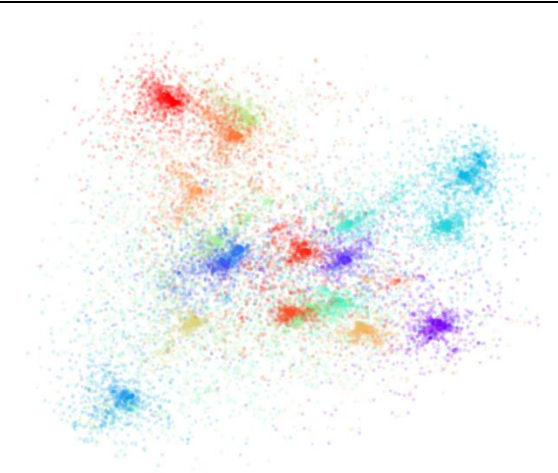

可以發現大多為正確的 **label**，但仍出現了較不重要的字如 **file**，可能是因為它經常出現，但又未太頻繁，因而被當作特徵字。它散佈在不同種類之中，可能影響分類的正確性。

在手動加入 **stopword** 例如 **error, file, data** 這些經常分散在不同種類中出現，且無意義的字之後，最常出現的字是：wordpress, oracle, svn, apache, excel, matlab, visual, bash, spring, hibernate, scala, sharepoint, ajax, qt, drupal, linq, haskell, magento, mac cocoa

可以發現已經都是正確的 **label** 字了

2. Visualize the data by projecting onto 2-D space. Plot the results and color the data points using your cluster predictions. Comment on your plot. Now plot the results and color the data points using the true labels. Comment on this plot.





兩張圖用眼睛可以觀察到明顯的 **cluster** 大約有 18 個，其他的可能是因為 **visualize** 做的不好，或是維度不夠將 **data** 區分開，或是 **data** 本身就很分散，導致一個 **cluster** 分布得很開，所以在圖上看不太出來。

My prediction	True label
	

<p>可以發現一個 cluster 除了會有明顯聚集的地方之外，其實還有很多資料點散佈在周圍，也會和其他 cluster 的資料點重疊，表示我的 prediction 做的並沒有很好。尤其是和 true label 比較過後，發現有些兩個靠得很近的 cluster 我並沒有把他們分對，而是將很多屬於其中一個 cluster 的資料點歸到另一個 cluster 去了。</p>	<p>可以發現 cluster 大致有明顯聚集的地方，邊界也比較清楚，雖然仍是有很多資料點散佈在周圍，但是和我的 prediction 相較之下並沒有那麼混雜。在兩個靠得很近的 cluster 處，仍可以區分出大致的邊界，沒有分的這麼開的原因可能是因為降維的緣故。</p>
---	---

3. Compare different feature extraction methods.

方法為 BoW/TFIDF*有/無 **stopword**，之後再使用 LSA 降維並用 Kmeans 分類：

BoW without stopword : 0.10725	TFIDF without stopword : 0.34009
	
BoW with stopword : 0.64895	TFIDF with stopword : 0.68230
	

由 Kaggle 分數以及分布圖都可以看出來 TFIDF with **stopword**>BoW with **stopword**>TFIDF without **stopword**>BoW without **stopword**。

也就是說，有無 **stopword** 的影響比有無使用 TFIDF 的影響來的更大。

例如在 TFIDF without **stopword** 時效果並不太好，可能是因為這些 **stopword** 出現

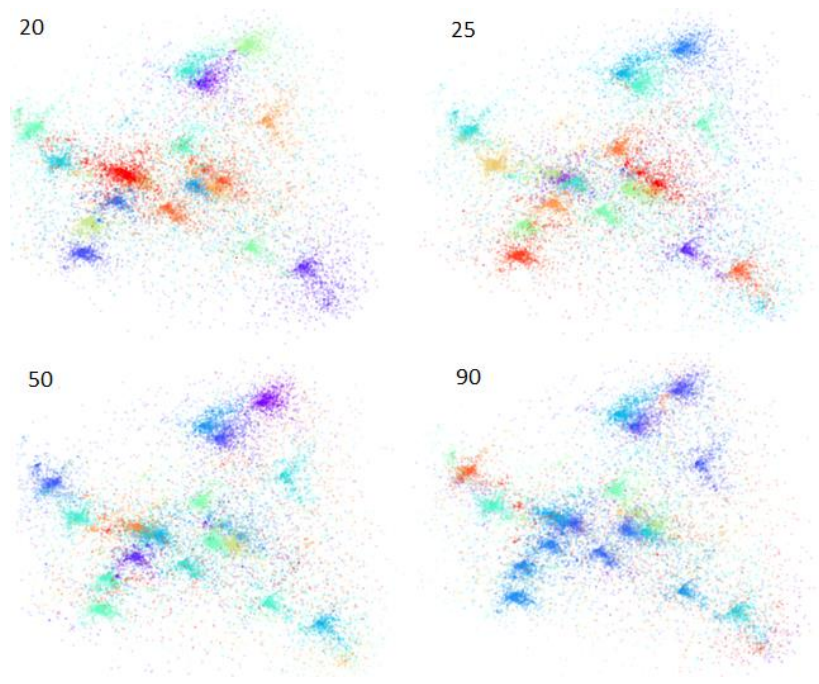
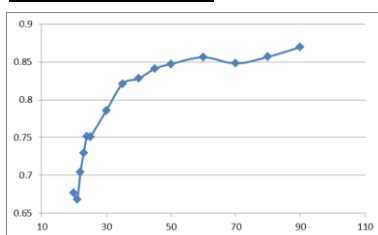
次數很多，也分散在不同 topic 之中，因此就算有使用 TFIDF，仍然可能沒有被剔除掉，而成為重要 feature 之一，因此造成 Kmeans 分類的困難。另外也有可能是因為 TFIDF 中可以調整參數決定要保留多少個特徵字，可能調大一些的話，能夠剔除掉比較多無意義的字。

另外，在 BoW with stopword 時，效果已經很不錯，可能是因為利用 stopword 就可以剔除很多常常出現但是不重要的字，甚至還比 TFIDF 做到的效果更好，而將不重要的字剔除後，剩下的字大多是重要的特徵字，直接拿來做分類而不加成，也可以達到不錯的效果。

當然，TFIDF with stopword 可以取到重要的字，且可以表現出這些字在文章中的重要性，因此效果最好，分布圖也最清晰。也可以發現 BoW with stopword 在兩個 cluster 的邊界處分的比較不清晰，但 cluster 大致都有出來。TFIDF without stopword 雖然大致的 cluster 有出來，但是周圍混雜了許多其他顏色的點點，分的非常不好，而 BoW without stopword 根本幾乎沒分出什麼來。

4. Try different cluster numbers and compare them. You can compare the scores and also visualize the data.

#	score
20	0.67689
21	0.66819
22	0.70458
23	0.72969
24	0.75189
25	0.75110
30	0.78583
35	0.82119
40	0.82807
45	0.84118
50	0.84730
60	0.85625
70	0.84809
80	0.85682
90	0.86977



發現分的 cluster 數目越多，Kaggle 分數越高，非常奇怪，因為分的越多就會把原本同個 cluster 的 data 分到不同 cluster 中，應該會變得比較不準。推測可能是因為 testing data 中大多都是不同 cluster 的 pair，所以才會這樣。但當我觀察 cluster 中 data 的數目，發現分成 20 個的時候每次 Kmeans 出來的 cluster 中 data 數目有很大的差異，有時候會有一類有 2000 個左右，有時候會出現的很平均，大約每類都 1000 個，而平均的時候分數也會比較高，應該是分的比較準。實驗後發現如果稍微提高 cluster 數目，可以將數目太多的那一堆分成較平均的兩堆，這樣真的也會讓結果更準確。從分布圖中也可以看出來 cluster 分的越來越細。