Student ID: R05944004

Name: Cheng-Wei Wu (巫承威)

hw 4
Unsupervised Clustering & Dimensionality Reduction
Report

- Q1

為了分析每一群中最常出現的詞，我先使用python的套件「nltk」來做stopwords的過濾，接著使用TF-IDF來抓出「title_StackOverflow.txt」裡面的特徵。抓出特徵之後，由於維數過大，因此再使用「Truncated Singular Value Decomposition (truncated SVD)/LSA 」來做降維。經過測試得知降到20維的時候效果平均較佳。最後再分別將真正的label跟我自己預測出來的分群結果的label同時套用在剛剛處理過後的檔案，利用「sklearn」裡面的「CountVectorizer」來分析每一群中每個詞出現的頻率，由於每個群的編號並不影響結果，因此我統計之後依照單字名稱排序一遍後再將結果用長條圖的方式呈現，以方便比對兩種label中每群最常出現的詞。結果如下圖所示：

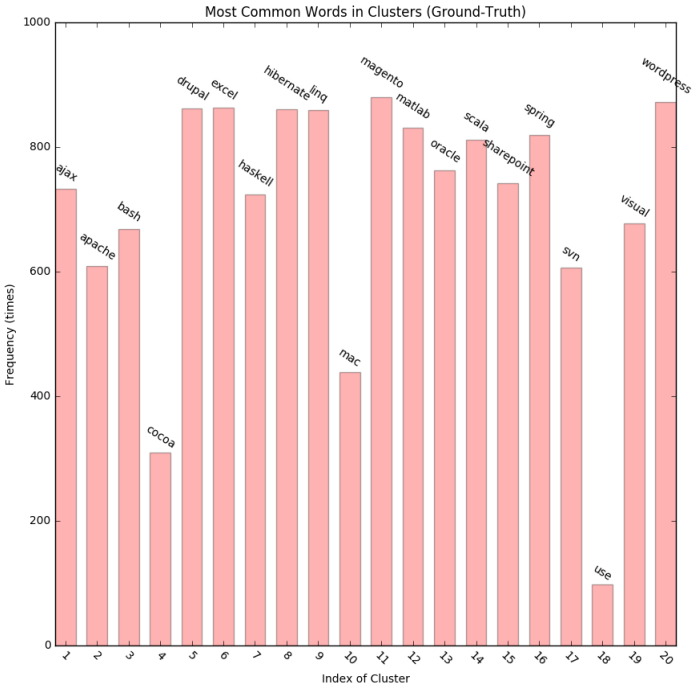

Figure 1.
Most Common Words of True Labels' Cluster

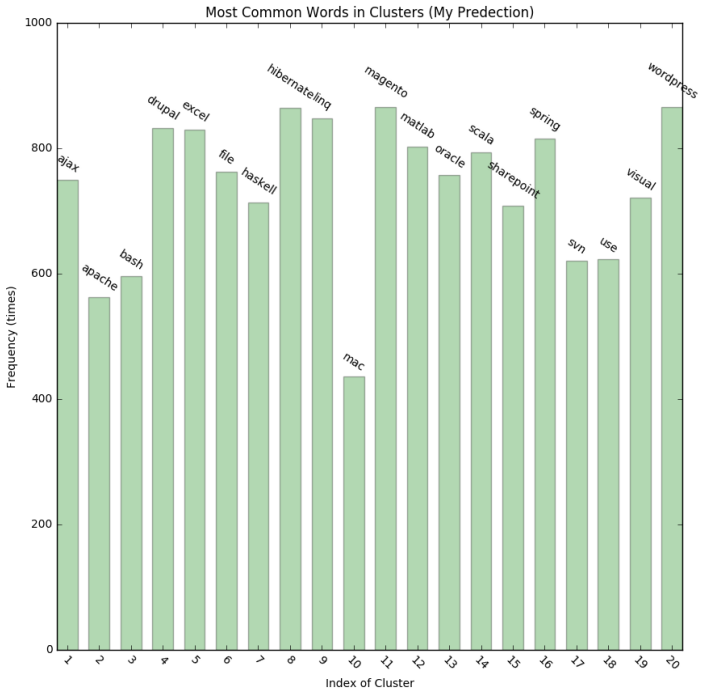

Figure 2.
Most Common Words of my Clustering Results

觀察後可以發現，基本上我預測出來的結果已經跟真正的label差不多了，每一群中最常出現的詞也幾乎一樣。根據上圖的結果，兩張圖中的群只有cocoa(file)這部分不一樣。而若跟投影片中提供的真正的tag比較的話，「wordpress, oracle, svn, apache, excel, matlab, visual-studio, cocoa, osx, bash, spring, hibernate, scala, sharepoint, ajax, qt, drupal, linq, haskell, magento 」其中因為「mac」跟「osx」其實算是一樣的類型，而「visual-studio」則因為連字號的關係所以剩下「visual」這個字，因此實際上其實只有兩個群的主要詞沒有跟tag對到，由此可以說明我分群的效果其實已經能約略表示各群中的特性，也就是每群中最常出現的詞。

- Q2

底下左圖為我自己使用如Q1的方式所畫出來的分群圖，右圖則是將「true label」套用在我已經分好的群上來畫點（圖中較大的圓型資料點為各個群的中心點）：

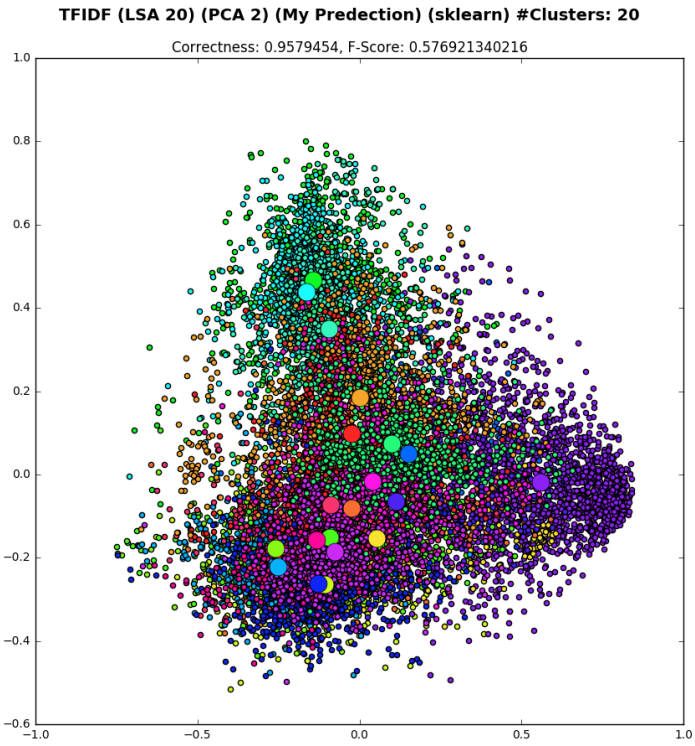

Figure 3.
Data Point Projected onto 2-D Space (My Prediction)

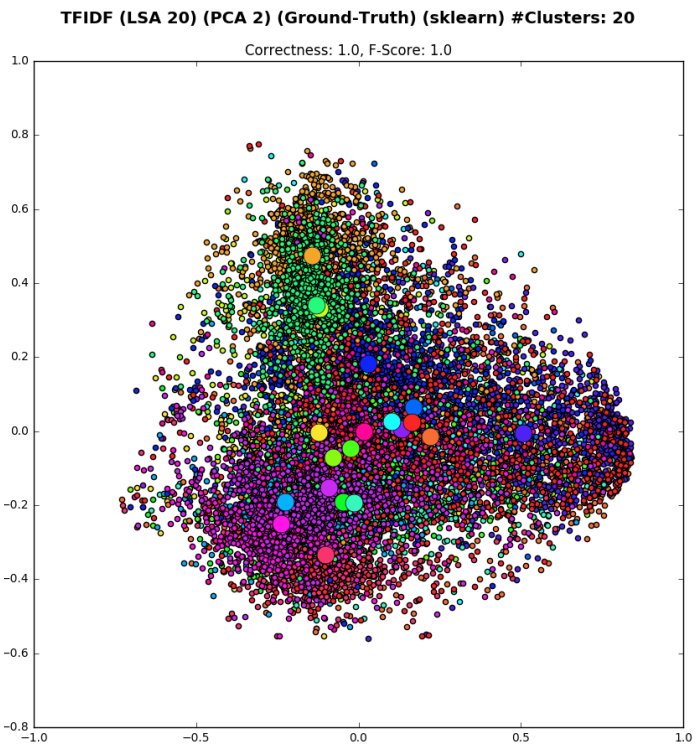

Figure 4.
Data Point Projected onto 2-D Space (Ground-Truth)

由於分出來的群是透過我用「K-Means Algorithm」分好的，兩者不同的點只有在每個資料點所屬的群不一樣的差別，所以基本上兩者的資料點的分佈算是非常相似。不過因為label不同的關係，所以還是會有些微的差別，因此可以很容易地發現兩圖中各個群的中心點分佈比起資料點的分布較為不同。

- Q3

我實作了用不同的降維方式「PCA」以及「SVD/LSA」來處理透過「CountVectorizer」與「TfidfVectorizer」從文件中抓取出來的特徵。另外，由於在一些情況下，「CountVectorizer」所計算出來的詞頻率特徵會比「TfidfVectorizer」算出來的特徵較能代表文件，但另一方面，「TfidfVectorizer」所提取出來的特徵卻較能代表詞的意義與獨特性，因此我多做將兩者合併後的特徵「CountVectorizer + TfidfVectorizer」進行降維來分群的部分。以下是分群完後使用true label計算F-Score的一些結果比較（上表格與下表格為分別使用與不使用「nltk」來處理stop-words的結果，評斷方式為F-Score）：

#Cluster = 100	CountVectorizer	TfidfVectorizer	CountVectorizer + TfidfVectorizer
PCA	0.661918070129	0.705483622485	0.69235750257
SVD/LSA	0.775795887771	0.764638896552	0.771309709585

#Cluster = 100	CountVectorizer	TfidfVectorizer	CountVectorizer + TfidfVectorizer
PCA	0.801255730715	0.838292723735	0.808665680448
SVD/LSA	0.84252056416	0.828784654175	0.848063966238

- Q4

根據Q3的結果，我用不同的組合方式中來比較不同群數的特徵抽取與降維方式，下列圖呈現了不同群數下的F-Score與將其資料點視覺化後的結果（礙於版面限制，故資料點視覺化的部分只列出Q3中平均表現最好的方式「CountVectorizer + TfidfVectorizer & SVD/LSA」）。

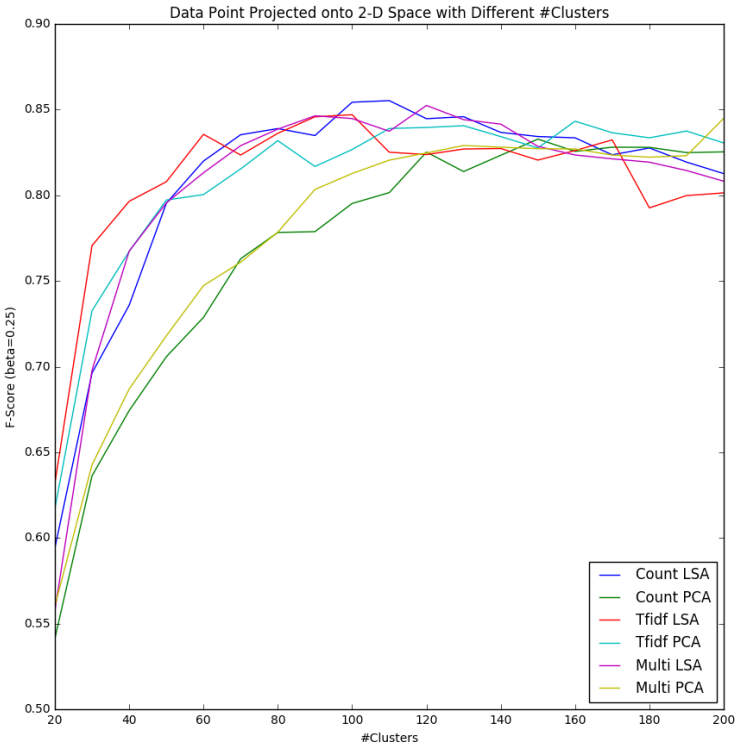

Figure 5.

從左圖可以發現，當群數從基本的20群開始增加到約略80至120群時，F-Score平均都有大幅度的增加。而從120群開始增加至200群時，使用PCA來降維的所計算出的F-Score平均而言仍繼續增加，但若使用LSA降維則會開始有下降的趨勢。對於增加群數使得F-Score大幅增加的原因，我推斷是因為在測試資料中，並非直接比對每個title所屬的實際tag為何，而是比較任意兩title是否為同一群。因此若當某兩資料點的位置接近於某些群的邊界時，群數小的時候會有較高的機率將此兩資料點群聚在同群；而群數大的時候則容易將此兩資料點分在不同的群中。基於這樣的情況，可能會有誤將兩點拆開或是誤將兩點合併的平衡風險需要考慮，而從左圖的結果觀察，可以得知平均而言，群數大的表現較群數小的為佳。

下列圖則呈現資料點在不同群數下的分佈：

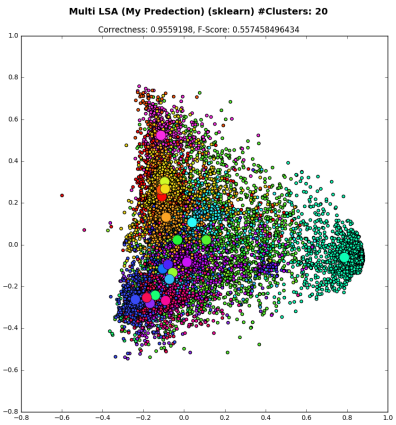

Figure 6.

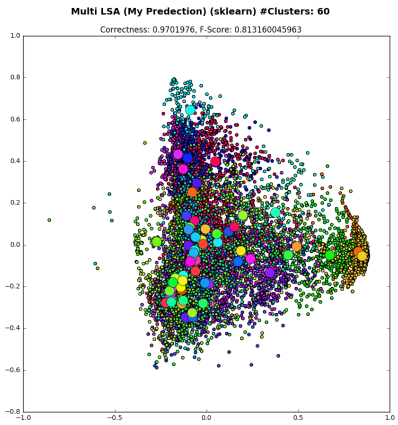

Figure 7.

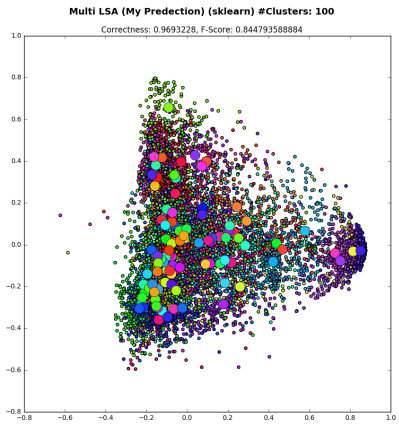

Figure 8.

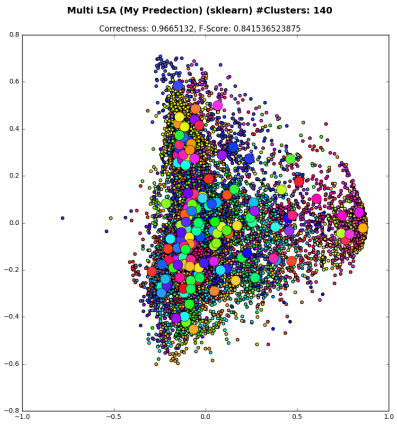

Figure 9.

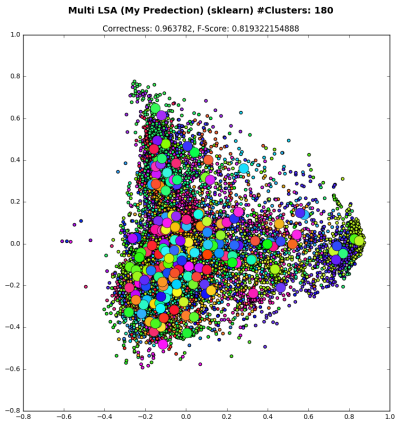

Figure 10.

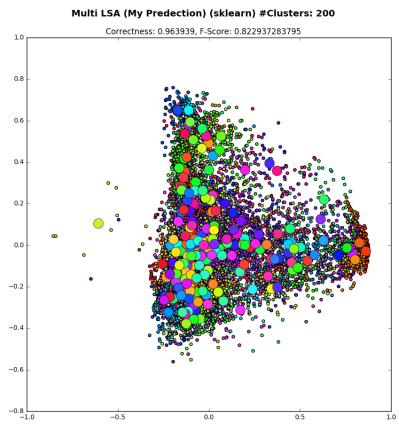

Figure 11.