ML Report4~ Unsupervised Learning

1. Analyze the most common words:

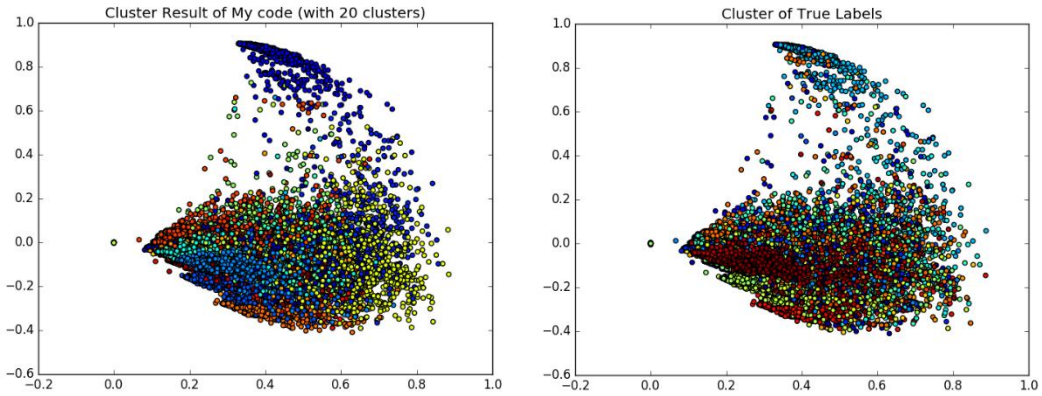
這題要我們找出各 Cluster 當中比較常出現的字。

我呼叫了 Sklearn 中 CountVectorizer()這個函式，其中我的 Stop words 使用的是內建的'english'這個 set，排除英文當中比較常見的字眼。我在 Cluster 完以後透過反推得到不同 Cluster 當中最常出現的五個字，將它們列在下表當中。我使用的是我自己 Cluster (TF-IDF, LSA, 20 clusters)後得到的 label，也就是我在做 K-means 時會使用的判斷依據，可能跟助教提供的 label 順序略有不同。

(0): file,qt,bash,excel,matlab	(10): hibernate,mapping,query,problem,criteria
(1): bash,script,command,shell,files	(11): wordpress,page,post,posts,ceremony
(2): oracle,sql,query,table,database	(12): visual,studio,2008,project,2005
(3): ajax,jquery,php,net,asp	(13): sharepoint,list,web,custom,site
(4): scala,java,type,use,class	(14): magneto,product,custom,products,page
(5): using,qt,druple,apache,list	(15): excel,vba,data,cell,net
(6): linq,sql,query,using,list	(16): matlab,function,array,matrix,image
(7): haskell,type,function,list,error	(17): mac,qt,os,application,file
(8): drupal,mac,os,use,custom	(18): spring,mvc,bean,security,web
(9): apache,rewrite,server,file,php	(19): svn,files,repository,subversion,server

2. Visualizing the Data

* Scatter Plot: Clusters of My Result / Clusters of True Label



* Comment on both plots:

這兩張都是利用 TF-IDF 加上 Truncated-SVD (LSA)去取 X-Y 軸的 feature。

首先看到第一張，也就是我的結果：因為我 Clustering 採用 K-Means 的關係，所以可以看到群聚在一起的點一定是屬於同一個 Cluster，而離各 Cluster 比較遠的點則會找尋較近的 Cluster 組去歸類。簡單來說，用 K-Means 做出來的結果，分群相當明顯，且例外不多。

而第二張，也就是 True Label 的真實結果，比較不同的地方在於，整張圖的顏色分布較亂，有些點離自己屬於的 Cluster 群體相當遠，這也就是整個 Data 當中比較偏向 Exception 的部分，明明同一個主題，其內容卻有滿多的不同。因為我們使用的是 K-means，不太能夠分辨這些例外點，而這應該也是我 Kaggle 上錯誤的主要來源了。

3. Comparism of Different Feature Extraction Methods

(1). Bag of Words (with LSA)

第一個做法是純粹的 Bag of Words，基本上就是把每個 term 數出來做為 feature，不做任何後續處理。我在這部分的 Vectorizer 當中，完全不加入任何的 Stop words 相關限制，所有 terms 的 weight 一律等價。

這個手法其實表現不差，可以過 Simple Baseline。

(2). TF-IDF (with LSA)

這個做法除了利用 Bag-of-Words 手法將 Terms 數出來，還會對每個 term 進行 weighting 的動作，此外也會移除一些比較常出現的文字。

我在這邊基本上就是增加了一些 Sklearn 函式使用，首先是我使用了 Sklearn 當中內建的'english'文字集合作為 Stop words。再來我使用的是，sublinear_tf 作為 TFIDF 進行 weighting 的工具。

我 Kaggle 上最好的成績便是由這個手法得到，關於成績會在後面進行比較。

(3). Auto-Encoder

第三個作法是使用 Auto-Encoder。

這個手法在上次的作業基本上已經做過，我是從上次的 model 改的。一樣是用 Sklearn 的 Vectorizer 取出陣列以後，利用 Keras 的 Dense()建立一個 Encoder 和 Decoder 組合，然後進行 training 動作。之後用 Encoder 取出的結果，搭配 Cosine Similarity 來進行 K-means 的 Cluster 動作。

這個手法表現滿差的，但上一份作業表現也頗差，所以不太期待它的結果。

(4). Word Vector (with TF-IDF, LSA)

最後一個做法是 word vectors。

我的方法是，利用 docs.txt 先去進行 vectorize，輸出一個 vocabulary 的字典，之後在對 Title_StackOverflow.txt 進行 vectorize 的時候，以這個字典作 input，只需考慮這些字典內容的字即可。簡單來說是利用 docs.txt，來決定 Title 當中有哪些 term 要被後續的 TF-IDF 考慮。

這個手法的表現中規中矩，沒有純粹使用 TF-IDF 來的好。

*** Performance Table (Kaggle private score)**

# Bag of Words only	約 0.49
# TF-IDF (Kaggle BEST!)	約 0.83
# Auto-Encoder	約 0.75
# Word Vectors	約 0.63

*** Comment**

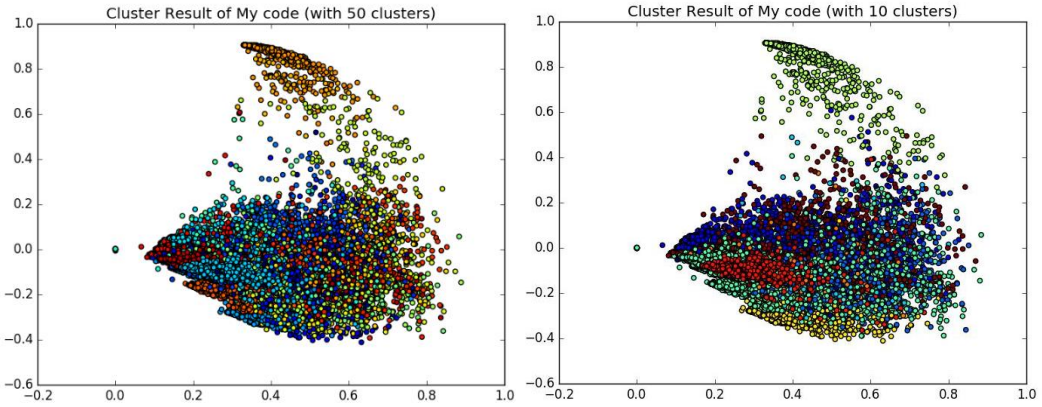
首先是純粹 BoW 以及 TF-IDF，這邊 TF-IDF 的表現較好，這是顯然的。因為 TF-IDF 對 Vectorize 的結果進行剪枝和加權處理，如此會使重要的 feature 能夠更顯著的影響最後 Cluster 結果—我們可以從結果看出，同樣的方法只是增加了 TF-IDF，成績進步了非常多。

再來先比較 Word Vectors 和 TF-IDF，TF-IDF 的表現也比較出色。這邊兩者的差距在於，TF-IDF 是直接使用 Title 的內容來進行 Vectorize 的動作；而 Word Vector 是依照該 term 在 docs.txt 出現頻率範圍為基準。直接使用 Title 的內容，在對 title 做 cluster 之後得到比較好的結果，想想也是意料當中的事情。(如果允許 Title 去 Match Document 那麼結果可能會不太一樣)

最後談談 Auto-Encoder，Auto-Encoder 其實算是在做 Dimension Reduction，使準確率下降是合理的表現，我們用本來處理好的結果通過 Encoder 後，維度下降而使 Clustering 的判斷變少反而變得不準確。

4. Comparism of Different Number of Clusters

我分別將 cluster 數量增加至 50 個和減少至 10 個。我將各自的 Scatter Plot 和 Kaggle performance 顯示在下面：



*** Performance Table (Kaggle private score)**

# 10 Clusters	約 0.588
# 20 Clusters	約 0.661
# 50 Clusters (Kaggle BEST!)	約 0.831

*** Comment**

這兩張圖片可以搭配上面的 20 clusters 那張圖觀看。我們可以發現，一點也不意外的，50 clusters 的 clusters 數目變多，每個 cluster 群落的分布範圍變小；10 clusters 的 clusters 數目變少，每個 cluster 群落的分布範圍變大。

然而，這邊的意義在於，由於我們的 test case 是要 identify 任意兩筆資料是否有在同一個 Cluster 當中。如果我們在乎的是判斷在同一個 cluster 當中的準確性，我們可以選用較多 cluster 數目；反之，如果我們在乎的是判斷不在同一個 cluster 的準確性，我們可以選用較少 cluster 數目。而從我的 Kaggle score 可以判斷出來，助教提供的 test case 計分方法，大概是偏向前者。