Machine Learning HW4 Report

b02502108
陶昇永

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

Ans:
當我沒加任何 stop_words 時，most common words in 20 clusters 如下：

wordpress, in, hibernate, excel, to, is, and, haskell, linq, matlab, with, magento, of, ajax, bash, visual, on, with, do, drupal

可以發現有多沒有意義的 tag，於是加入預設 stop_words， 'english' ，其 most common words in 20 clusters 如下：

svn, drupal, qt, linq, bash, using, magento, file, scala, wordpress, excel, hibernate, matlab, apache, spring, visual, sharepoint, haskell, ajax, oracle

可以發現已經好很多了，大部分的 tag 已經和投影片提供的相同，但仍有出現 'file' 等不具意義的 tag，這是因為本次的 tag 大部份跟程式有關，故 'file' 也大量出現，但卻散佈在各個 tag 中，造成 cluster 失準，故我們再手動挑除一些 stop words 加入，其結果如下：

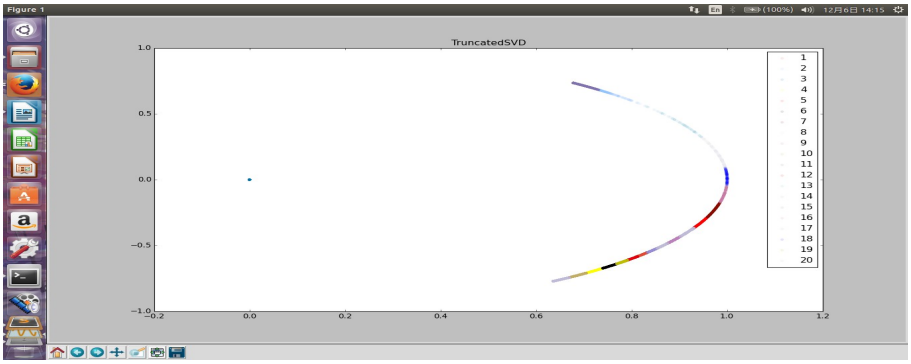
wordpress, linq, sharepoint, scala, visual, apache, cocoa, drupal, matlab, ajax, hibernate, bash, oracle, excel, haskell, svn, spring, mac, magento, qt

發現已經與投影片提供的 tags 相同！故採用此組 stop words 來當作 tokenize 的基礎。

額外 stop words 取法：先將 title tokenize 之後, 取 DF 大於 100 的字出來看, 若是無意義的字, 或是會廣大分散在各個 cluster 的字,就將他加進 stop_words 中, 重新做 tokenize, 以免之後在做 truncated SVD 的時候, 會考慮到這一個沒有太大意義的維度。舉例像是 data 這個 word, 就有可能出現在表列 20 種 tag 中, 故刪除。

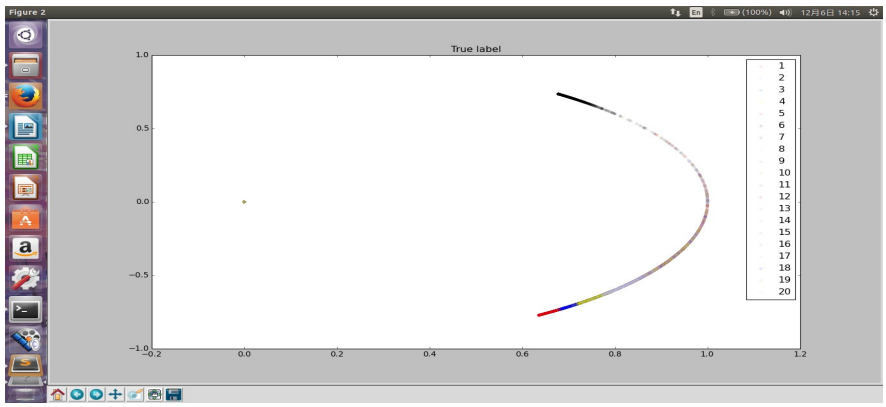
2.Visualize the data by projecting onto 2-D space. Plot the results and color the data points using your cluster predictions. Comment on your plot. Now plot the results and color the data points using the true labels. Comment on this plot.

Ans：
a. using cluster predictions(TruncatedSVD with normalization):



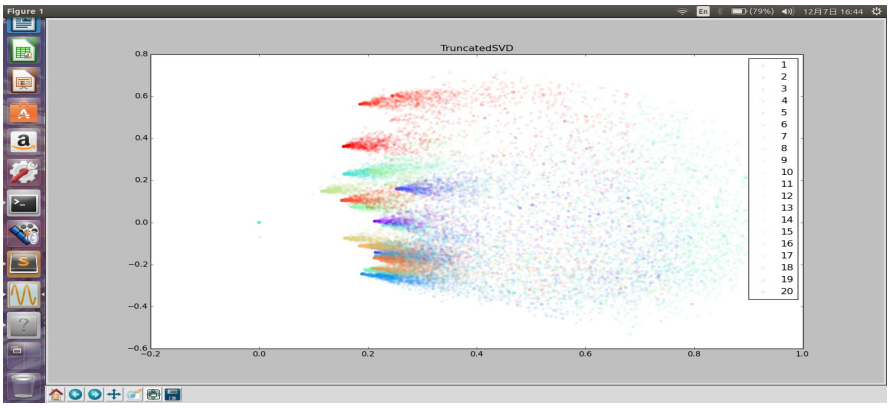
經由標色，我們可以發現曲線上每一段都代表一種 cluster，所以大致上是有成功的將資料 cluster 出來。

b. using true labels(TruncatedSVD with normalization):

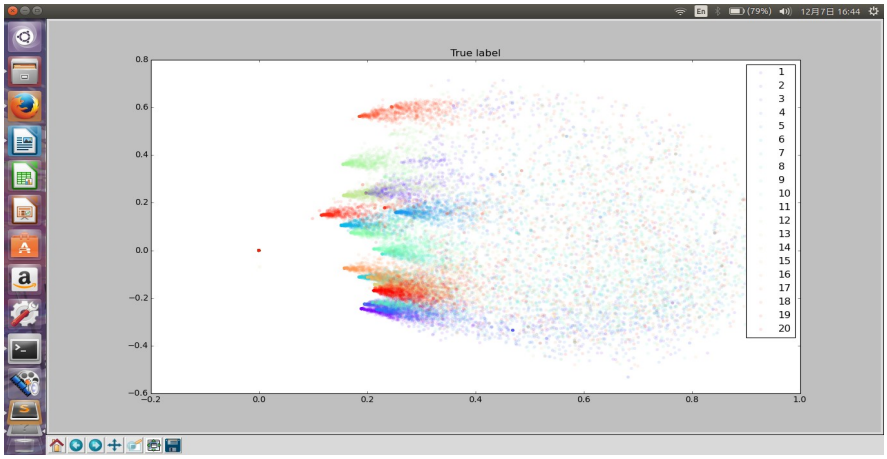


由此圖與 a 圖比較，可以觀察出只有前端和尾端有真正的分對 cluster，在曲線中間顏色都是灰色，為各色的點混雜產生，表示用 TruncatedSVD 降成二維的作法，並無法將 data 分成原本的 20 個 tags。

c.using clustering predictions(TruncatedSVD without normalization):



d. using true labels(Truncated without normalization):



可以看出當沒有做 normalization 時，圖形散佈的比較亂，但仍看的出各 cluster 分開的趨勢(左方開頭處)。

3. Compare different feature extraction methods.

Ans:

方法 a. Bag of Words，且不加 stop_words:
Kaggle performance: 0.36143
方法 b. Tf-Idf，且不加 stop_words:
Kaggle performance: 0.682
方法 c. Bag of Words，且加 stop_words:
Kaggle performance: 0.686
方法 d. Tf-Idf，且加 stop_words:
Kaggle performance: 0.717

可以發現，當使用 bag of words 且不加入 stop_words 時，效果其差無比，經過 TF-IDF 和 stop_words 的修飾後，整個表現上升許多。

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

Ans:

	Kaggle performance
20 clusters	0.609
21 clusters	0.686
22 clusters	0.717
23 clusters	0.727

另外，我也發現，當我分成 100 clusters 時，kaggle 分數居然會衝到 0.85，可是這感覺十分的不合理，分太多 cluster 應該會把同 tag 給分開才對，這不是我們想看到的，也不應該會很高分。所以我覺得可能是 kaggle 的 test data 有點不平均，才會導致此現象。