Unsupervised Clustering & Dimensionality Reduction

b02901014 王崇勳

1. Analyze the most common words in the clusters.

根據 TF-IDF 可以找出在每個 cluster 裡面出現最頻繁的字如 Table.1。

excel	use	apache	drupal	linq
hibernate	spring	visual	sharepoint	wordpress
bash	qt	matlab	svn	magento
ajax	mac	oracle	haskell	Scala

Table.1

根據投影片裡面提供的 tags 可以發現自己找出來的 tags 多了 use、mac，少了 osx、cocoa 其中 visual-studio 可能是因為在拿掉標點符號的過程中被分成 visual 和 studio 才會造成找出來的 tags 不同。

2. Visualize the data by projecting onto 2-D space.

此部分都是先利用 TF-IDF 取 feature，再利用 LSA 降到 20 維，最後用 kemeans 分成 20 個 clusters。

首先將 20000 個 features 降到 2 維，然後根據 kmeans 的 label 給予不同的顏色並畫在 2D 平面上，如圖 Figure.1-1 與 Figure.1-2。從圖中可以看出自己 cluster 和 true label 的分布，兩者的 clusters 呈現一段一段分布，整體外形也差不多。

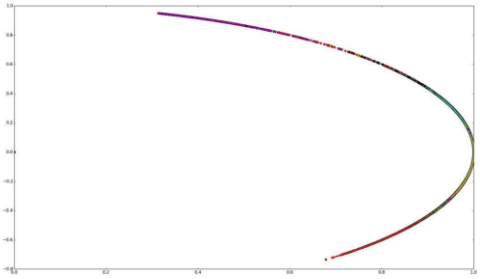

Figure.1-1 my cluster

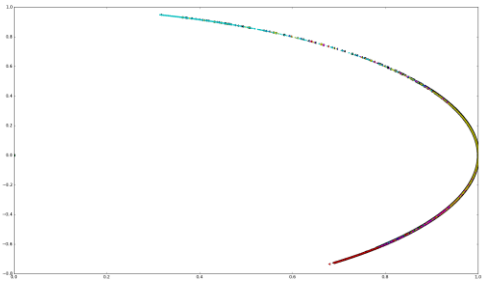

Figure.1-2 true label

接著嘗試先將 20000 個 features 降到 3 維，然後取 feature 的第一與第三 elements 畫到 2D 平面上，如圖 Figure.1-3 與 Figure.1-4。從 Figure.1-3 中可以比 Figure.1-1 更清楚看出 cluster 的分布。然而 Figure.1-4 卻顯得有些凌亂。原因可能是因方面從高維度投影在 2D 平面上，cluster 與 cluster 間本來就有可能重疊；另外使用 kmeans 分 cluster 時是利用幾個 means 當作中心分群，分出來的 cluster 可能比較容易聚成一個”球型”，投影在 2D 上面可以比明顯看出分布。

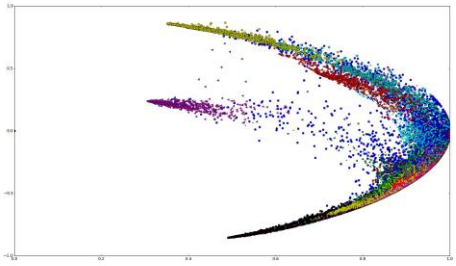

Figure.1-3 my cluster

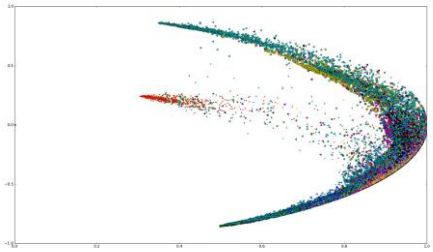

Figure.1-4 true label

3 Compare different feature extraction methods.

Bow	TF-IDF	Stop words	LSA (n_components=20)	Feature size	score
V				(20000,11110)	0.06634
V		V		(20000,10876)	0.13611
V			V	(20000,20)	0.10098
V		V	V	(20000,20)	0.56101
	V			(20000, 5450)	0.30793
	V	V		(20000, 5236)	0.18526
	V		V	(20000,20)	0.33468
	V	V	V	(20000,20)	0.63188

(1.) Bow 只單純統計一個 term 出現的次數，而 TF-IDF 除了考慮一個 term 出現的頻率外，還會考慮一個 term 具有的類別區分能力(IDF)。因此使用 TF-IDF 取 feature 效果較好。

(2.)在 feature 維度較高時，使用 stop words 雖然能夠減少維度，但是比例很小，對於分群的結果影響不大。但是使用 LSA 降維到 20 以後，有沒有使用 stop words 就有明顯的差異。原因可能是若沒有把 stop word 拿掉，降維過後會得到 with, of, in, on, to, is, for, and, do 等字，跟原本的 tags 有很大的差異。

4 Try different cluster numbers and compare them.

Cluster number	score	Cluster number	score
20	0.63188	80	0.85210
40	0.82422	100	0.85250
60	0.84772	120	0.84279

Table.2

根據結果可以發現當 cluster number 為 100 時分數最高。原因可能為當芬比較多 cluster 時，原本屬於同一群但被分到不同群的 feature 可能會被分到同一

群；然而原本分在同一群的 feature 也會被分開。因此應該存在一個 cluster number 使分數最高。這次我所使用的方法，這個 cluster number 就是 100。最後嘗試使用 cosine similarity 比較兩個 feature 相似程度。結果如 Table.3。

Keans (n_cluster=100)	Cosine similarity (>0.9)	Score
V		0.85250
	V	0.89931