Machine Learning HW4 Report

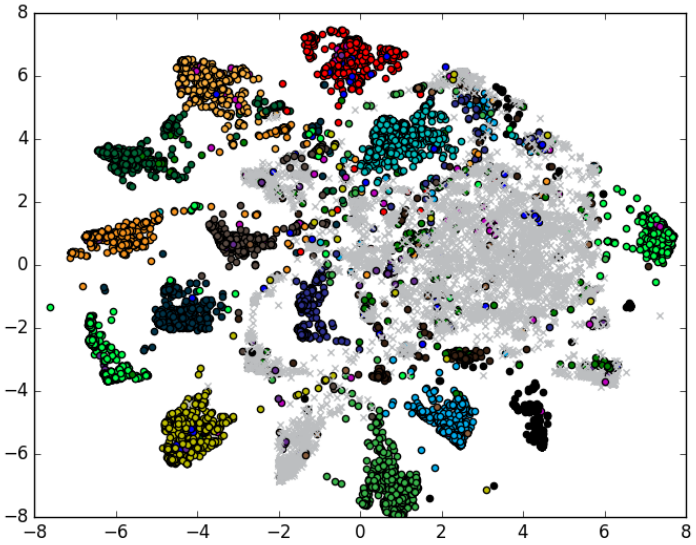
R05944013 網媒一 高滿馨

1.下表列出title的所有單字中，最常出現的20個字，經過TF-IDF計算過後的值的range

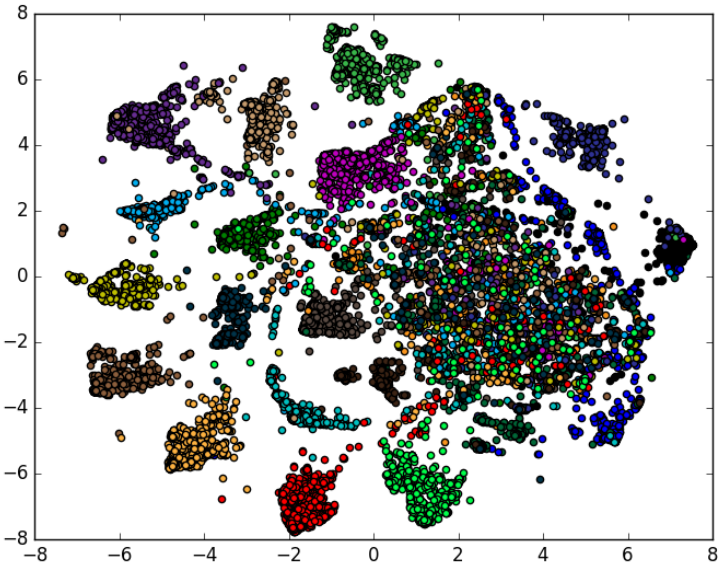
an	0.14~0.31	excel	0.14~0.52	how	0.10~0.18
and	0.12~0.33	for	0.12~0.34	in	0.07~0.28
can	0.14~0.31	from	0.09~0.40	is	0.12~0.22
do	0.14~0.29	hibernate	0.17~0.40	magento	0.15~0.52
of	0.13~0.33	on	0.13~0.35	the	0.11~0.32
to	0.09~0.30	using	0.15~0.30	what	0.14~0.33
wordpress	0.13~0.51	with	0.10~0.28		

可以看出一些很常出現但沒什麼意義的字(the, an, of, to …) ，最高的數值都不會超過0.35，而一些比較有意義的字(wordpress, magento, hibernate)等，最大值都比較高(0.5左右)，因此只需設定一個threshold，值低於某個threshold的word就不考慮，這樣就可以過濾掉許多沒有意義的字。

2.



Visualization with cluster prediction



Visualization with label

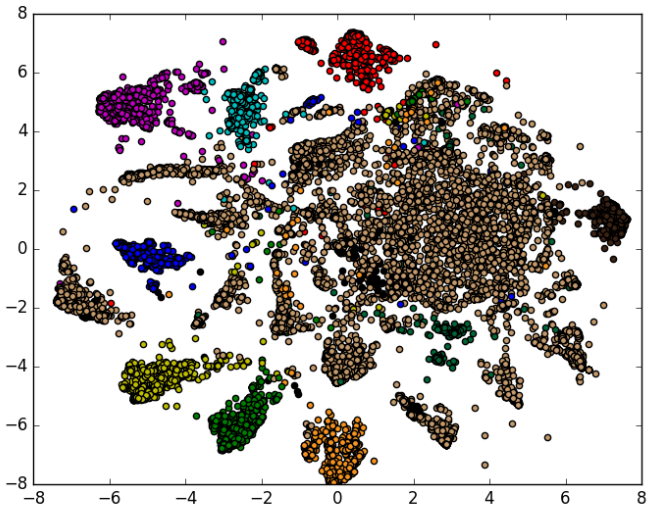
可以看出兩邊的cluster的結果大致相同，左邊prediction的部分，灰色叉叉的部分是分不出來的data，對應到右邊就剛好是一塊很混亂很多混合tag的區域，所以無法有效的分辨出來。

3.
Method1: Bag of words
以單字出現的次數當做feature，有做過前處理，直接把stopwords拿掉，然後選擇全部出現次數最多的500個單字來當作feature。
Method2: TF-IDF
先用bag of words計算出每個字出現的頻率，然後再去計算TF-IDF值當作feature。
Method3: Bag of words with LSA
一開始都與方法一相同，但做完之後會再使用LSA把維度降到100維。
Method4: Train word vector
因為方法一到方法三的方法都只透過“同樣的字”出現的頻率來計算feature，但是其實有很多字雖然不同，但是有同樣的意義，所以想試著用doc來train word vector，來學習相似字，這樣就可以處理用字不同，但意思相同的title。有使用genism的library來train出一個word vector，並列出每個tag的前10個最相關的字，但發現計算出來的word vector每個tag都會互相有關聯，因此預估做出來的效果不會太好，所以就沒有加上去。
example tag: wordpress (後面的數字為相關程度, 越高代表越相關)
[('Wordpress', 0.8909342885017395), ('**magento**', 0.8677185773849487), ('**Magento**', 0.8467340469360352), ('sharepoint', 0.8180790543556213), ('WordPress', 0.817542552947998), ('Drupal', 0.8139870166778564), ('drupal', 0.8078259229660034), ('SharePoint', 0.783298671245575), ('theme', 0.7816714644432068), ('blog', 0.7622872591018677)]
example tag: magento
[('Magento', 0.8720846772193909), ('**wordpress**', 0.8677184581756592), ('drupal', 0.8664113283157349), ('sharepoint', 0.8644723296165466), ('**Wordpress**', 0.8494300842285156), ('Sharepoint', 0.8422864675521851), ('**WordPress**', 0.826980471611023), ('Subversion', 0.8171219825744629), ('Drupal', 0.8143742680549622), ('subversion', 0.813186526298523)]

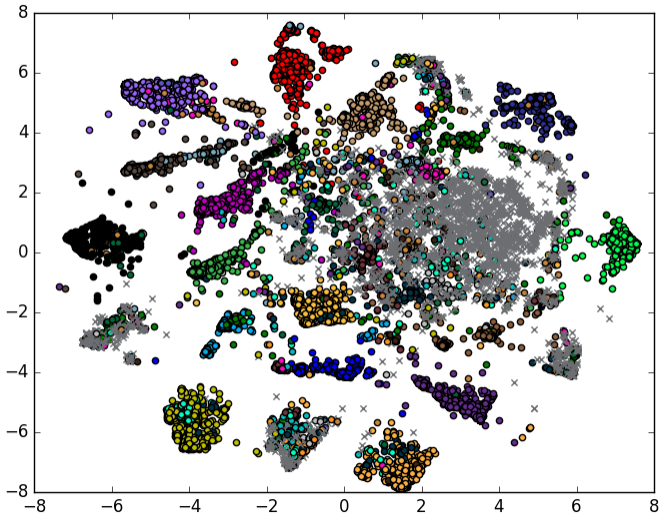
最後發現效果最好的取feature方式是先對文字做好前處理(轉小寫, 拿掉標點符號、去除stopWords之類的)，再取BoW當feature。後面clustering的部分是用K-means，實作方法是去取20個互相垂直的feature當作K-means的center point，再做clustering。另外新增一個全部都是0的維度，來處理那些完全分不出來的class。這樣去做K-means clustering，

不做LSA降維的效果會比做LSA降維的效果好。可能是因為有去指定center點，所以效果比較好。另外TF-IDF的效果比起BoW的方法不是很理想，可能因為資料量不夠，有些stopwords沒有辦法很有效地去除，所以導致最後效果不好。

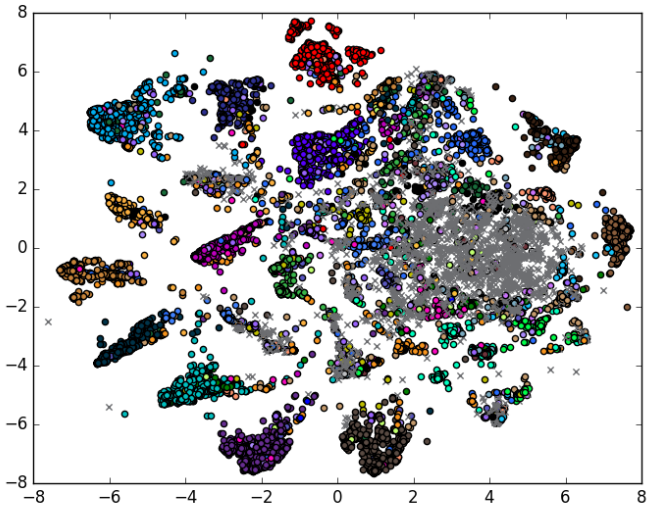
4.



cluster number = 10



cluster number = 30



cluster number = 40

沒有實際去測試準確度，但從圖上可以看出，cluster number越多，那塊混亂區域的面積比較小，看起來比較整齊，因此做出來的結果理論上會比較好。

討論人: 鄭嘉文