ML HW4

B03902071 葉奕廷

1.Analyze the common words

首先我找 common words 的方法是將各個 cluster 的 stop-words 移除後用 TF 對每個 word 做 weighting 並印出 weight 前幾名的單字。

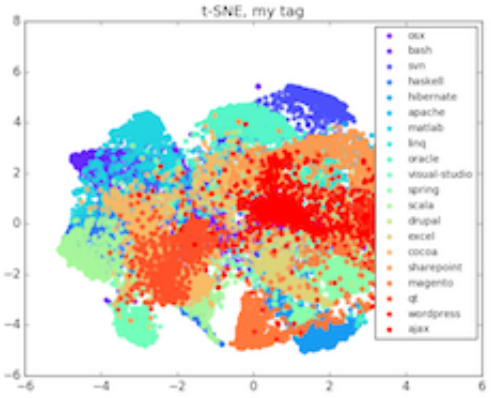
很明顯的幾乎每個 cluster 中出現頻率最高的 word 就是該 cluster 的 tag (有 osx 的 cluster 'mac' 最常出現這個例外)。還有很多 cluster 出現頻率高的 word 是很好用常識去預測的，像是 oracle 的 cluster 中的 'sql' 、haskell 的 cluster 中的 'type' , 'function' 及 bash 的 cluster 中的 'script' , 'shell' , 'commmand' 出現頻率都很高。而有些對我來說比較意外的像是 apache 的 cluster 中 'rewrite' 頻率非常高，這應該代表用 apache server 的人出現的難以解決問題有很大一部分和 rewrite 相關，還有像 visual-studio 的 cluster 中 '2008', 'project' 經常出現，或許代表 visual-studio 2008 最多人用 or 問題很多，然後問題常出現在開 project 上。

還有像是 'file' 在 svn, bash 兩者的 cluster 都是熱門出現單字，或許我們可以用 'file' 先去分出屬於這兩個 cluster 的 title。還有像是 '2007' 這個字常出現在sharepoint 的 cluster，我們搞不好還可以先用 title 中是否出現西元年來先區分該 title 的 tag 是不是屬於微軟的產品，因為微軟的產品很喜歡用西元年去取名。

2.Visualize the data

在此我使用 t-SNE 來將我的 data 投射到 2-D 平面上，因為我原先的 feature 最後有 60 維有點大，所以我先用 LSA 將 feature 降到 10 維才做 t-SNE 以減少計算時間。

t-SNE,true tag 是用 true label 來塗色的圖，t-SNE, my tag 是用我自己的 prediction 來塗色的圖，我自己的 prediction 的 tag 是我把各 cluster 的 title 輸出後人工去 tag 的。(各 tag 對應的顏色在兩張圖不同)



在我自己的 prediction 的圖上很明顯同樣顏色的點幾乎都是聚集在一起一群一群的，這代表 kmeans 確實有將 feature 相近的 title 分到同一個 cluster，而細看可以發現有些同樣顏色的點是零散的，我想這是 feaeture

降維時造成的誤差。

接著我們可以發現在 true tag 的圖上有非常多同樣顏色的點是散落在各處的，這並無法純用降維的誤差來解釋，而是意味著有很多 title 我取出的 feature 雖然並不相似，但它們其實是該被分到同一個 cluster 的。值得慶幸的是在圖上還是可以看出有一塊一塊相同顏色的點聚集在一起，這代表我仍是有正確的 clsuter 大部分的點，像是圖上的右下和最左邊兩張圖的顏色分佈幾乎是一樣的，我的 feature 在這些地方對應的 tag 表現的非常好。

3.Compare different feature extraction methods

在 feature extraction 上我主要嘗試過兩種方向 --- tfidf 與 wordvector，以下將分兩部分對這兩個方向的細節做討論。

3.1 tf-idf

tf-idf 相信是一個最直觀的方法，然而若是只將各 title 換成小寫並將 stop-words 除掉做 tf-idf 在 kaggle 上的準確度只有 0.24 多，是個蠻差的結果，我想是因為純粹做 tfidf 得到的 feature dimension 太大，導致 feature 中無用的雜訊太多而無法很好的代表每個 title 。

若是在 tf-idf 後加上 LSA 去降維可以明顯的提高 cluster 的準確度，將 LSA 的 dimemsion 設為 20 的話準確度大概為 0.62，相對於沒做 LSA 的結果準確度提高了 40% ，這應該代表我們經過 tf-idf 得到的 feature 其實用 20 維就足以表達了，改用更精簡的 feature 能使最後 cluster 的結果得到很大的改善。

在 tf-idf + LSA 的組合我還試了不移除 stop-words 會不會比較好，因為雖然根據常識 stop-words 並無法帶給我們什麼資訊，但是我想搞不好因各個 tag 的發文者組成的不同會導致發文習慣的不同，所以某些 tag 的 stop-words 比例特別高也說不定，然而最後的準確度只有 0.34 ，無用的 stop-words 資訊反而降低了最後 cluster 的準確度。

3.2 Word Vector

除了 tf-idf，我還嘗試去 train 一個 word to vector 的 model 然後將每個 title 中的 word 的 vector 去取平均，以此作為該 title 的 feature。

我的 w2v model 的 feature dimension 為 500 並 train 了 25 個 iteration, window size = 5，有預先把 training data 全部換成小寫且移除 stop words。

Train w2v model 的演算法我選擇了 skip-gram 因為我發現用它比用 CBOW 的準確度高了 1~2%，我想這是因為 stackoverflow 這種偏技術的文章用幾個關鍵字去推測上下文的能力更能展現出對關鍵字的了解，而關鍵字的 feature 好壞影響最後的準確度很大。

純粹使用 word vector 我得到準確度 0.57 ，相對於純粹使用 tfidf 的 0.24 有了蠻大的進步，而若是用 word vector 取完 feature 後再用 LSA 降維降到 dim = 60 的話準確度可以進步到 0.65，同樣驗證了適當的降維可以提高準確度。在此我還有嘗試用 PCA 同樣降維到 dim = 60 然而準確度只有 0.58 ，我想這是因為 word vector 比較偏向 sparse data 所以 LSA 的表現會比較好。

接著我還嘗試將 window size 提高為 20 並仍是用 LSA 降維，準確度進步到 0.72。將 window size 提高到 20 代表一個單字幾乎要預測一整句話，這樣子使得 train 出來的關鍵字的 feature 更有用，使得最後的準確度有明顯的提高。

4 Try different cluster numbers

我首先嘗試了將 cluster number 減少為 10 並傳上 kaggle，準確度由 0.72 降為 0.32，用猜的也知道 cluster number 少於 tag 數的話一定會分錯非常多，準確度變成 1/2 一點也不意外。然後試著將 cluster number 增加為 40 沒想到準確度變成了 0.73，我想這是因為如果將 cluster number 增加可以使每個 cluster 內的 title 的相似性又更高，適當的增加 cluster number 可以將我原本錯以為在同一個 cluster 的 title 分開，使得準確度增加。

接著我利用 pca 降維並 visualize 了 data，可以發現在 cluster number = 40 時和 cluster number = 20 時其實沒有很大的差別，整體的 cluster 分佈仍是差不多的 ，代表如果 feature 夠相似 cluster 的結果是不會被稍微增加的 cluster number 所影響，只是在一些模糊地帶又分的更細了。