Machine Learning Assignment 4

Tung-Chun, Chiang R05922027

December 4, 2016

1 Data Preprocessing

We only use all titles removing all non-alphabet symbols(including numbers) and goes through the following three processes:

1.1 Camelcase parsing

We observe that the titles are all about CS, which usually contains functions named by Camelcase. We should separate it, so that the split words are more possible to match the target words, e.g., "$VisualSVN$" $\rightarrow$ "$Visual$" "SVN" and the svn is the target words.

1.2 Stop word removal

Stop word removal is the most important part in data preprocessing. We remove the words which have lower information to reduce the noises. The methods to remove those words are many; A simple way is to use the stop word lists which come from linguistic analysis, but they are general methods. In fact, stop words change with different corpus, e.g., "x" means nothing in general, but in the corpus in this assignment, "x" may means "x" in "os x" and it should not be removed. Removal by TF-IDF may also not a good way because the corpus is "titles", and people usually write titles in short sentences, which contain the meaning people want to ask. So the important words must appear with high frequency, which leads low TF-IDF. Finally, we count the frequencies of all words and remove some common stop words such as "i", "have", "am", and some prepositions. The Table 1 shows the effect of stop word removal, lists the top 10 frequency words and average TF-IDFs of them. The average tf-idf of "to" and "in" is larger than "vba" in cluster_1, which is the important word in cluster_1, so as "in", "to", "a" and "plot" in cluster_3. So we should not use TF-IDF to remove stop words in this task.

1.3 Stemming

Stemming is the process of reducing inflected(or sometimes derived) words to their word stem. This method reduces the vocabulary size and focuses the information of words. In this assignment, we use Porter stemming in all words after lowercasing them.

2 Clustering

Figure 1 is clustering results colored with predictions and Figure 2 is the results colored with true labels. This clustering use mean of word2vec to represent titles and cluster them with KMeans

cluster_1				cluster_2				cluster_3			
origin		removed		origin		removed		origin		removed	
word	ave. tf-idf	word	ave. tf-idf	word	ave. tf-idf	word	ave. tf-idf	word	ave. tf-idf	word	ave. tf-idf
excel	4.87	excel	4.53	linq	4.67	linq	4.35	matlab	4.80	matlab	4.47
to	1.32	vba	1.02	to	1.61	sql	1.13	in	1.76	plot	0.65
in	1.30	cell	0.89	a	0.94	queri	1.07	to	1.03	function	0.47
a	1.00	file	0.47	in	0.82	by	0.43	a	1.05	matrix	0.62
how	0.89	data	0.47	sql	1.20	join	0.47	how	0.87	array	0.49
the	0.65	macro	0.54	queri	1.14	group	0.45	of	0.77	imag	0.46
vba	1.08	sheet	0.54	how	0.72	list	0.36	the	0.44	vector	0.51
cell	0.94	row	0.44	use	0.73	select	0.42	plot	0.68	file	0.24
from	0.57	rang	0.43	with	0.69	data	0.34	and	0.40	d	0.34
an	0.60	column	0.37	i	0.54	tabl	0.35	function	0.50	find	0.21

Table 1: Some clustering results and the words before stop word removal

for 22 clusters. In Figure 1, Red cluster in right-down and orange cluster in left-top are garbage clusters because the corresponding labels are very impure. cluster of label ”qt” is not very good because ”qt” is the software development framework and cover many domains.

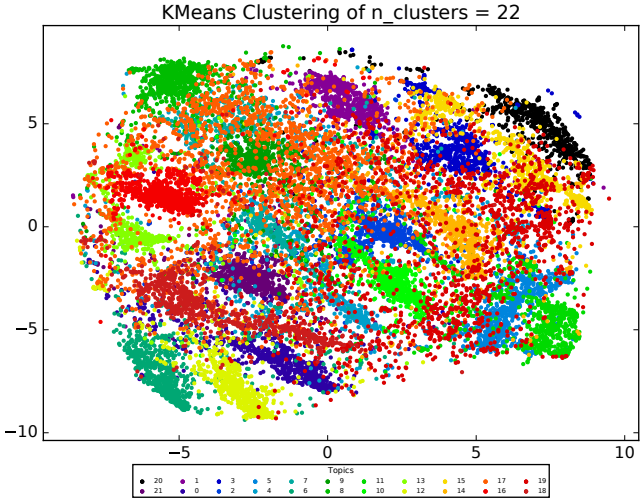

Figure 1: Clusters with prediction color

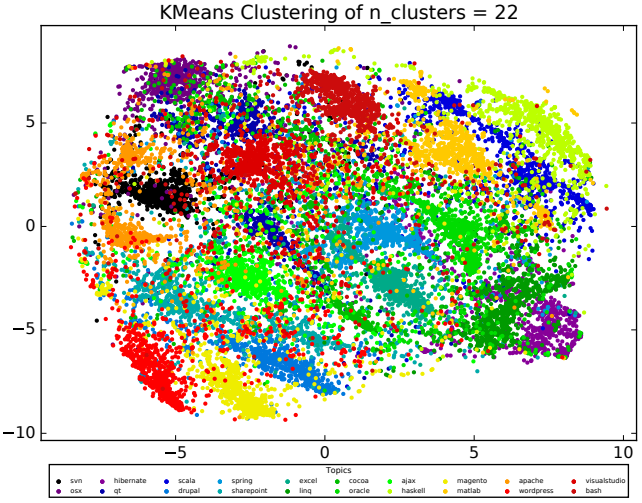

Figure 2: Clusters with ground truth color

3 Feature Extraction

3.1 Methods

All the following feature extraction methods use KMeans for 22 clusters.

- BOW: use only top 1000 frequency words, so the feature vector is 1000 dim.
- TF-IDF: the feature dimension is the same as BOW, but combine with IDF features.
- NMF: use non-negative matrix factorization to reduce TF-IDF vectors to 20 dim.
- Word2Vec: use only titles as corpus to train word vectors(300 dim, window size=3), and use mean of word vectors as document vectors.

3.2 Comparison and Analysis

The BOW outperforms TF-IDF method, as mentioned in stop word removal section, because TF-IDF does not represent documents well in this corpus, the titles. The NMF method reduce the sparse TF-IDF vectors to dense vectors and finds the correlation between words. Different from TF-IDF and BOW, which must match the words, NMF method matches the correlations of words, so performs better than TF-IDF and BOW. Word2Vec method consider only neighbors rather than the whole document. This implies the position of words(compared to BOW, TF-IDF, NMF, which do not care about the position of words), so the Word2Vec method performs best. The results are shown in Table 2.

methods	Kaggle score
BOW	0.22919
TF-IDF	0.19741
NMF	0.50812
Word2Vec	0.74723

Table 2: Kaggle score($F_{0.25}$) with different feature extraction methods.

4 Number of Clusters

Figure 3 and Figure 4 show the clustering result in different cluster number, colored with true labels. There is no obviously difference between the results. But when num_clusters changes from 20 to 22, the performance gets better. Because KMeans clusters the data points together that are hard to cluster into the 20 clusters. And the true labels in this kind of cluster are disordered, but it helps the others clusters to get more pure data. When num_clusters changes from 22 to 60, the performance gets worse. Because data in the same origin cluster are split into different clusters. If we don't merge those clusters, we may claim that they are not in the same cluster and make mistakes. The results are shown in Table 3.

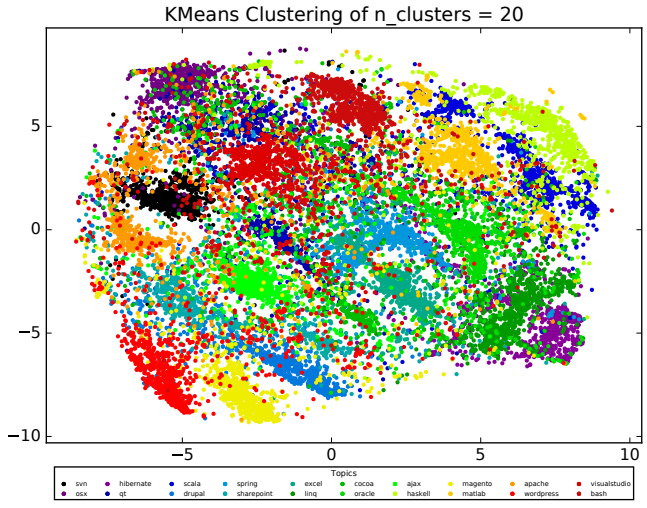

Figure 3: Clustering result with 20 clusters

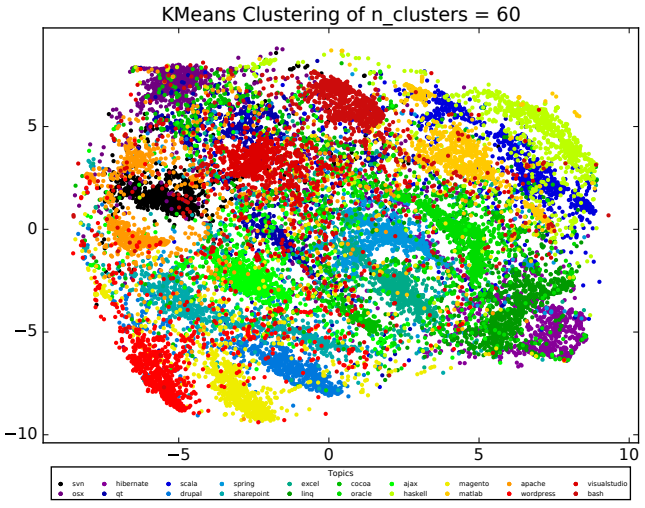

Figure 4: Clustering result with 60 clusters

num_clusters	20	21	22	23	24	25	60
Kaggle score	0.72575	0.71093	0.74723	0.73060	0.74141	0.73150	0.61812

Table 3: Kaggle score($F_{0.25}$) with different number of clusters.