1. Analyze the most common words in the clusters. Use TF-IDF to remove irrelevant words such as "the". (1%) (Note that this part was done by setting k = 20 in k_means for a proper analysis. However, my kaggle_best uses k = 53.)

#	Cluster id	#	Cluster id
1	wordpress	11	spring
2	hibernate	12	visual
3	matlab	13	linq
4	scala	14	using, file, drupal, excel
5	sharepoint	15	mac, os, cocoa
6	apache	16	ajax
7	excel	17	drupal
8	magento	18	Haskell
9	oracle	19	bash
10	qt	20	svn

Table I

We can see that except cluster 14 and 15, the other ones did a pretty good job separating the tags. The reason why cocoa couldn't be properly distinguished might because its occurrence is only 356 in title_StackOverflow.txt. On the other hand, cluster 14 seems to be holding all the other titles that can't be properly distinguished. For example, drupal occurred 870 times in the text, and excel occurred 894 times. Therefore, cluster#14 holds 1877/20000 of the tiles, which is the largest portion of all.

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 your plot. (1%)

The plots were done by using my best feature extraction technique, but with cluster number = 20, and using t-SNE embedding technique to visualize into 2-D.

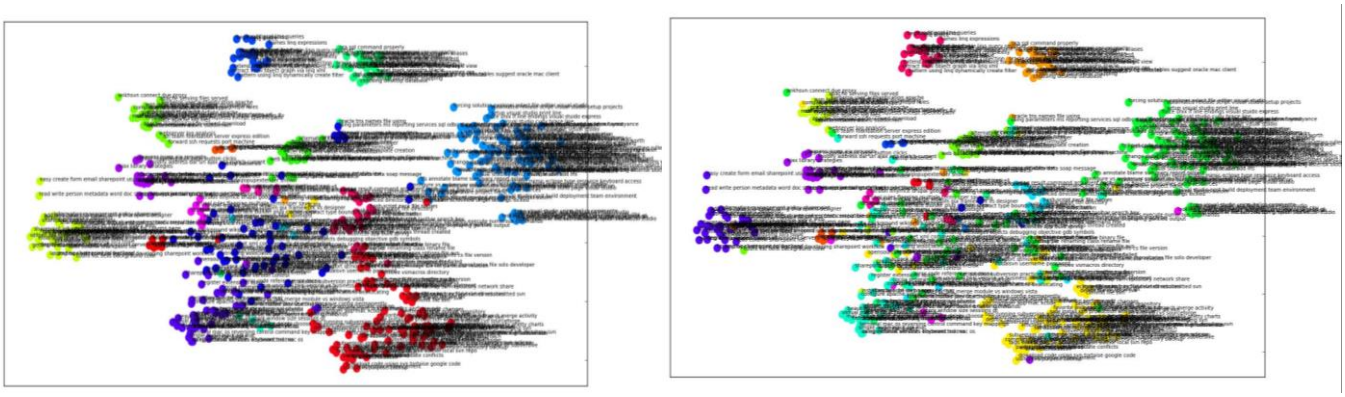

Fig.1 Plot by using only 500 titles, so that titles could still be visible
Left: my cluster prediction, Right: true labels

Comments:

From Fig.1, we can see that most of the obvious clusters could be clustered together, and the labels indeed are mostly correct, except in the center part of the figure.

To have a more comprehensive view of the result, Fig.2 is plotted by using 15000 titles. The whole set

couldn’t be used due to memory issues of t-SNE.

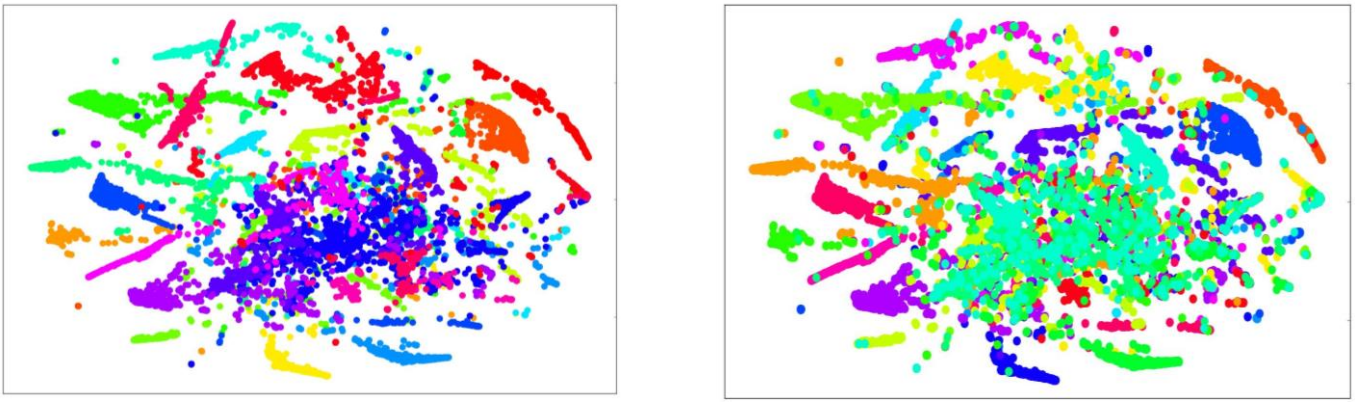

Fig.2 Plot by using 15000 titles, without showing titles

Left: my cluster prediction, Right: true labels

From Fig.2, we can obviously see that the light green color in the center seems to be prevalent over the plot. Therefore, it would get confused with other clusters.

3.Compare different feature extraction methods (2%)

(In this section, cluster numbers k is set to 20, instead of kaggle_best k = 53)

#	Methods	Private Score
(1)	BoW	0.158
(2)	BoW + TF-IDF	0.271
(3)	BoW + TF-IDF + LSA	0.334
(4)	BoW + TF-IDF + LSA + text preprocessing	0.655

We can see that BoW alone merely passes weak baseline by a slim margin, and the gradual application of TF-IDF and LSA indeed improves the performance each with a few %.

After using LSA, the text preprocessing is really beneficial to the result, as it almost doubled the private score. It was done in the fashion:

1. Eliminate stopwords.
2. Replace punctuation marks with white spaces
3. Conversion from upper-case to lower case

---

4. Try different cluster numbers and compare them. You can compare the scores and also visualize the data. (1%)

Cluster #	Private Score
18	0.578
20	0.655
30	0.792
50	0.832
53	0.844
55	0.841

We can see that the best cluster # is about 53. Based on my observation, I think the reason why >>20 cluster numbers achieves better results is because of the F-measure, and check.csv is very sparse, which means that most of the values in output.csv would be zeros. Therefore, the best way to achieve a high Kaggle score would be to “decide carefully” whether the 2 titles are in the same cluster or not. This could be realized by making the cluster # go further beyond 20.

In Fig.3, both cluster#20, and cluster#53 is plotted by using only 500 titles, and the labels are colored using my labels.

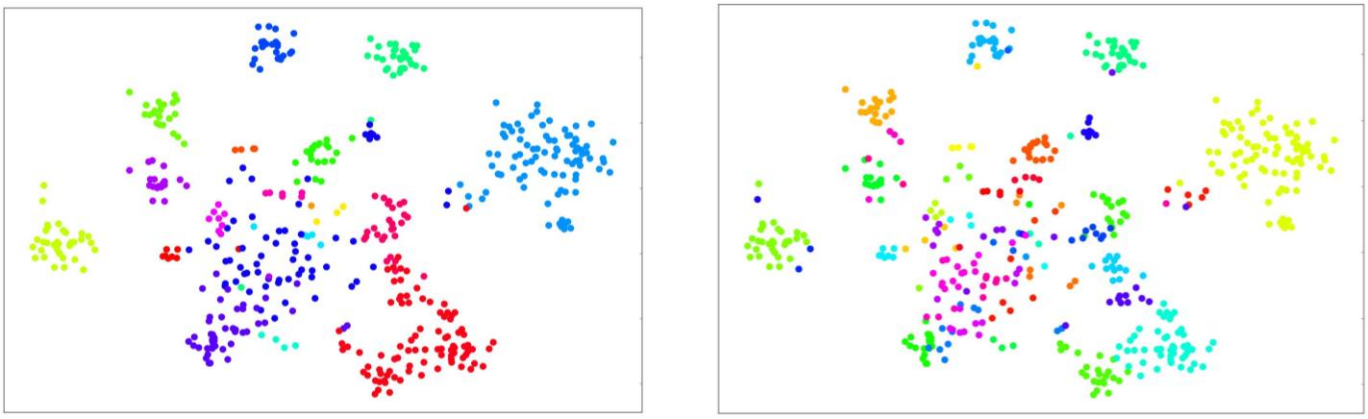

Fig.3 Left: cluster#20, Right cluster#53

From Fig.3, we can see that increasing the cluster#, we can further distinguish small clusters from the large ones. Therefore, by doing so, we could have a better F-score measure.