Machine Learning Homework 4 --- Unsupervised Clustering & Dimensionality Reduction

Author: b03901036 陳柏文 Collaborator: b03901037 鄭宇强 b03901137 張致綱

1. Analyze the most common words in the clusters. Use TF-IDF to remove irrelevant words such as “the”. The most common word is “the”. There are 56636 “the”s in total in “docs.txt” and “title_StackOverflow.txt”. The following table tells the top 20 frequent words in the text files. Basically, we can found that most of the words in the table are article, pronoun, preposition, verb to be, conjunction, auxiliary verb and possessive adjective. So I think that we can preprocess the documents by removing those words belong to the above categories.

words	**Part***	**Total****	**idf(t)*****	**words**	**Part***	**Total****	**idf(t)*****
the	article	56636	3.079892	this	pronoun	13824	5.386085
I	pronoun	47499	2.458614	for	preposition	12288	3.421419
to	preposition	47164	2.238614	use	noun/verb	11834	3.84392
a	article	44194	3.907771	have	verb	10891	5.483003
in	preposition	24846	2.268917	with	preposition	10088	3.30815
is	verb to be	21405	3.529536	on	preposition	9604	3.57773
and	conjunction	21149	3.413567	how	adverb	8912	2.608239
of	preposition	17681	3.316227	can	auxiliary verb	8756	3.942242
it	pronoun	15458	4.746559	do	verb	8266	3.75992
that	pronoun	14337	4.976612	my	possessive adjectives	8141	4.952895

*Part: 詞性

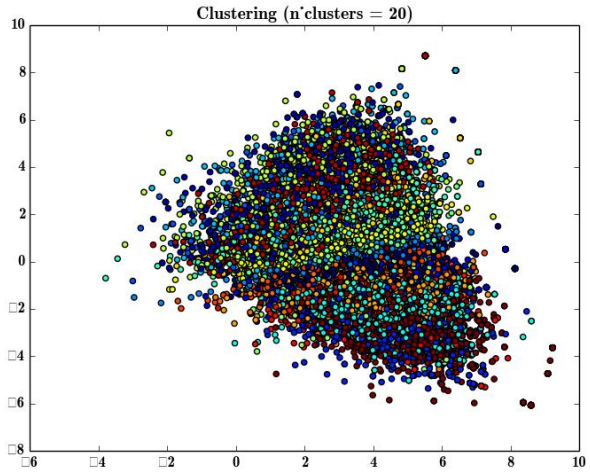
**Total: Total counts for each words in docs.txt and title.txt

***idf(t): inverse document-frequency based on the title.txt only and those rare words are assign the value: 10.210390. Thus, I think that TF-IDF can’t really remove the irrelevant words.

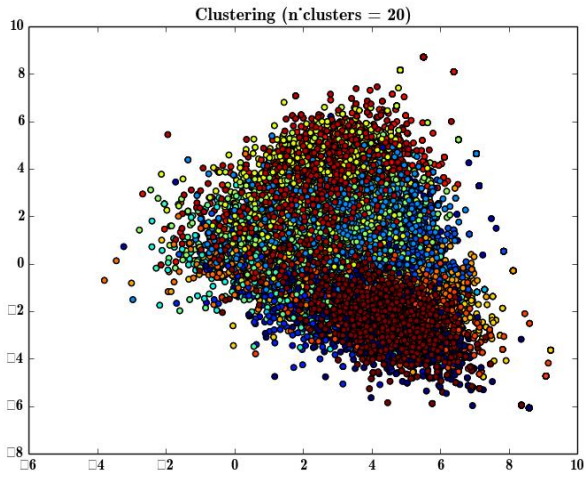
$\mathrm{idf}(t) = log\frac{1+n_d}{1+\mathrm{df}(d,t)} + 1$,where n_d is the total number of documents, and $\mathrm{df}(d,t)$ is the number of documents that contain term t. I think removing stop words is still needed, for that those most frequent words have large inverse document-frequency.

Cluster 0: mac cocoa os
Cluster 1: magento product products
Cluster 2: ajax jquery request
Cluster 3: hibernate mapping spring
Cluster 4: sharepoint list web
Cluster 5: bash script command
Cluster 6: scala haskell python
Cluster 7: excel vba data
Cluster 8: magento default wordpress
Cluster 9: visual studio 2008
Cluster 10: scala java type
Cluster 11: linq sql query
Cluster 12: wordpress drupal plugin
Cluster 13: subversion repository merge
Cluster 14: spring oracle svn
Cluster 15: linq sql spring
Cluster 16: haskell type function
Cluster 17: qt window windows
Cluster 18: matlab matrix vector
Cluster 19: apache server php

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 this plot.

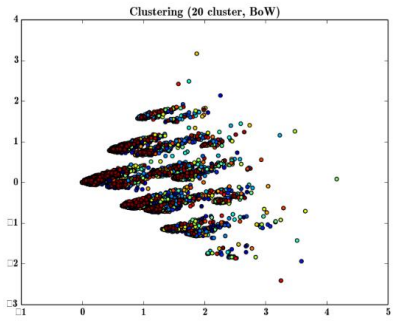


The picture on the right side is based the following method. I use Porter Stemmer and WordNet Lemmatizer in the nltk toolkit to stem and lemmatize first for all the documents. And I filter out stop words based on the list on the internet and some most frequent words. I use Word2Vector as my method for word embedding. And for training Word2Vector model, I call the library “genism”. Least but not less, I use truncate-SVD as my method for data samples visualization.



The below picture is based on the true labels. We get an accuracy of 0.7769 for the above model, but We found that the picture is really different. We guess that it is because the dimensions for each documents are far from 2, so when we project all the data sample on 2D plane. We get them stacked over each other. But we can still see some similarity, we can the yellow points mostly are distributed on the top half and the red points mostly are distributed on the bottom half. So we might conclude that they map to same classes.

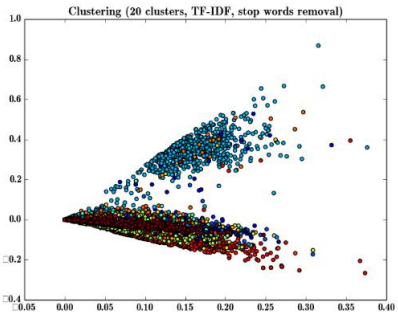
3. Compare different feature extraction methods.
   a. Bag of words (with stop words): baseline modal



We got an accuracy of 0.2265 based on the bag of words method. I think that bag of words, undoubtedly, is the worst modal. Since it didn’t remove those frequent words and the vectors that representing those documents actually are really sparse. Only partial columns are nonzero.

   b. TF-IDF (with stop words)

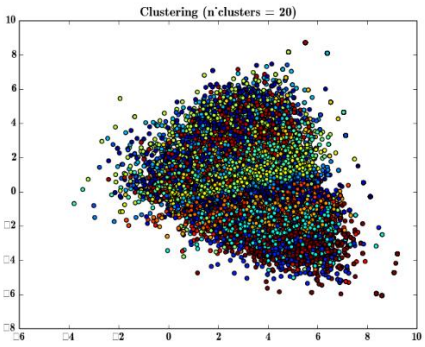
      We got an accuracy of 0.5785 based on the TF-IDF method.



c. TF-IDF (without stop words)

We got an accuracy of 0.6294 based on the TF-IDF method. Since for TF-IDF method, those frequent and meaningless words such as “the”, “are” and “is”, still have large inverse document-frequency. Thus, those words still have big impact on the performance. Adding stop words is still needed.

d. Word Vector (with stop words): best modal



We got an accuracy of 0.7769 based on the word 2 vector method. And the details are described above.

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

   In this section we compare the result of kmeans with TF-IDF vectorizer (filter of stop words applied) for feature extraction, given different numbers of clusters. Concluded from the following table, we find that adding more cluster will give better performance. I think the reason is that we can get more detail and the data sample are actually too divergent. So we have to add more clusters to avoid such different samples being cluster together. But adding too many clusters might get worse performance, since too many clusters will cause the same result as not cluster at all.

Clusters	5	10	20	30	50	**60**
accuracy	40.58%	55.18	80.22%	89.60%	92.57%	**94.06%**

Table: Different number of clusters versus accuracy

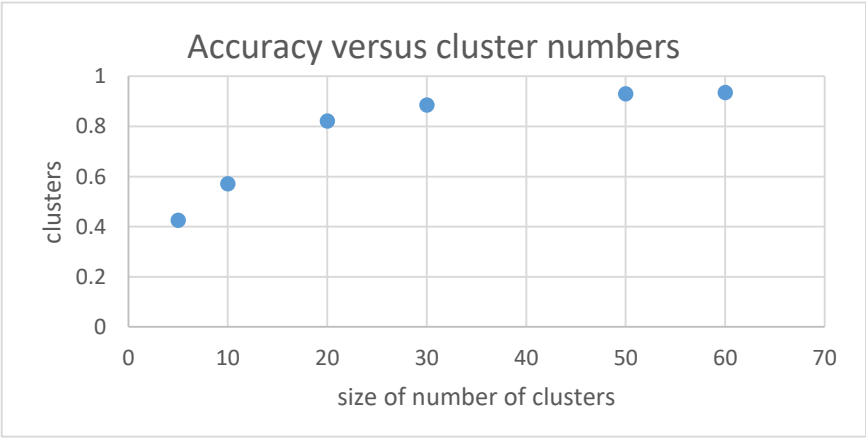


Figure. The plot of Table