EE219 PROJECT 4

Unsupervised Clustering Algorithm

Guanchu Ling 904590047

Yingchao Tang 404592020

Yang Guo 304681050

Introduction

In project 2 we did before, we tried to classify the textual data into different groups using the method of supervised classification such as support vector machine, Naive-Bayes classification and Logistic Regression and evaluated the performance of those methods. In this project, we turned to apply the unsupervised classification method on the same case. And we tried the K-means clustering method with different dimension reduction, optimized the dimension reduction separately and compared the performance with each other. We started with 2-cluster case and then moved on to the multiple cluster case and gave a detailed analysis on each of the outcome results.

Question (1)

In this part of the project, we did the almost same pre-process on the dataset: retrieved the documents with the offered API, removed the punctuations, headers, footers, quotes and stem words from the textual files, count and vectorize the files into a term-document matrix and convert the vectorized matrix into a TF-IDF matrix. So far we got an TF-IDF

matrix of size 7882 X 5245 which contains the TF-IDF scores of 5245 terms over 7882 posts.

Question(2)

After the pre-processing, we found that the TF-IDF matrix is still a sparse matrix containing much less valid information comparing with its actual size. And in this part, we tried the K-means(2-cluster) clustering algorithm on the raw TF-IDF matrix and evaluated the clustering performance by inspecting the confusion matrix as well as the homogeneity, completeness, adjusted rand and adjusted mutual scores of the labels predicted by the clustering algorithm. And here are the results:

```
Homogeneity: 0.44710281308
Completeness: 0.468635319246
Adjusted Rand Score: 0.49645964039
Adjusted Mutual Info Score: 0.447052194323
Confusion matrix:
==============
[[3873  106]
 [1058 2845]]
==============
Homogeneity: 0.41172421212
Completeness: 0.440314869403
Adjusted Rand Score: 0.44358645193
Adjusted Mutual Info Score: 0.411670353976
Confusion matrix:
==============
[[  90 3889]
 [2677 1226]]
```

As clustering algorithm is method of unsupervised learning, we combined all the data into one set and applied the K-means clustering. Different from the evaluation we used from the previous supervised learning, what we want now is now get the precise label-

consistent, but just a good separation of the data that is originally from different class, which means the clustering performance could be seen as good as long as the larger values in the confusion matrix lay on the same diagonal. In our result shown above, we could see that, the four scores are almost in the range between 0.41 and 0.50 (best should be 1), the larger two values in the confusion matrix lies on the same diagonal, which mean our algorithm clustering the most part of the data nodes to correctly. And one more thing we should notice here is that, the clustering algorithm is not very stable. The two sets of result we represented above are based on the exact same dataset. But from the confusion matrices of the two sets, they seem to be both diagonal but not in the same one, while the clustering scores are pretty much the same.

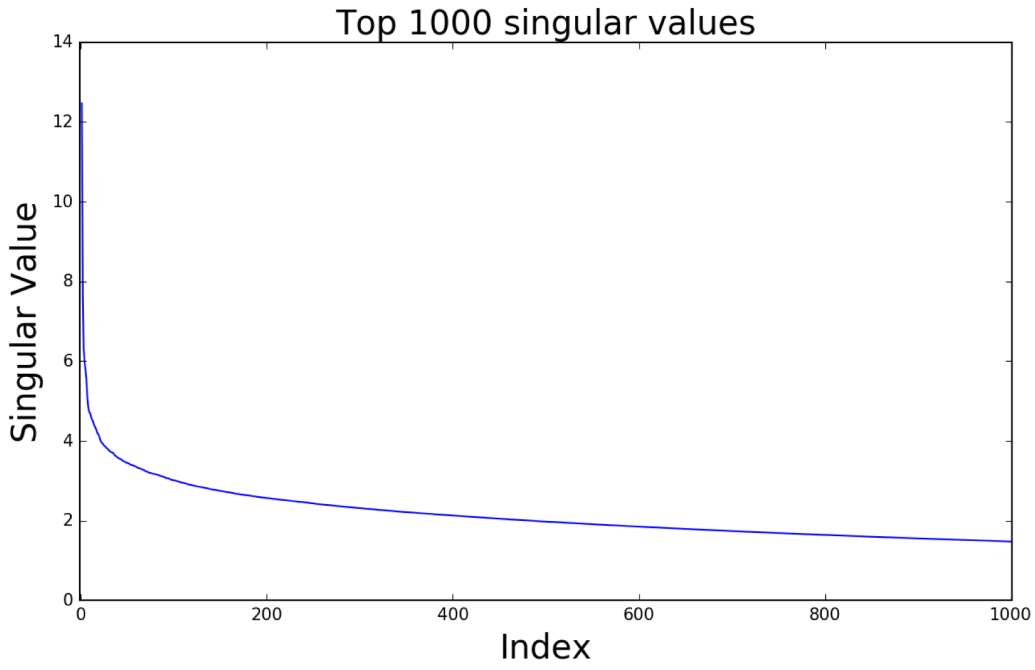
At the same time, we should notice that the falsely clustered data nodes are not yet minority, the clustering performance should still be improved in the following parts. The bad clustering performance could be resulted from the highly sparse matrix. While the built-in K-Means learning method from scikit-learn kit has been designed to take the sparse matrix, but without the dimension reduction, the signal-to-noise ratio of the sparse matrix would be very high. So as what we did in project 2, we tried to reduce the dimension of the TF-IDF matrix with different methods to see if the clustering performance would be better.

Question 3

As we can see from the previous results, although the confusion matrix looks like a diagonal matrix, the precision of the clustering labels is not as satisfactory as we expect it to be. This is possibly caused by high dimension TF-IDF matrix. Because we use the word count information to build the TF-IDF matrix, with high dimension TF-IDF, too many unimportant words would result in very serious overfitting, and data points is possibly to be distracted far away from where they should be.

To overcome this shortcoming and improve the performance of our algorithm, we will try to reduce the dimension of the TF-IDF matrix. By reducing the dimension, some unrelated information could be eliminated after the process (for example, by doing projection process). Only the most significant dimensions remain and ideally we can improve the precision of the cluster results. In this project, we will be using Latent Semantic Indexing (LSI) and Non-Negative Matrix Factorization (NMF).

By applying LSI, this technique takes into consideration the fact that words used in the same context tend to have similar (or close) meanings. LSI can help us discover the inner relation between different words. It is based on the singular value decomposition of matrix. Let D be the TF-IDF matrix, then the SVD of the matrix could be expressed as , where U and V are both unitary matrices and Σ is the corresponding singular value matrix. By doing this, we are able to get all the singular values of the TF-IDF matrix and get a reasonable guess on the dimensionality to feed in the K-means algorithm. We calculated the top 1000 significant singular values and plotted them as below:



In the plot above, singular values have sorted in descending order. As we can see, the decreasing rate (i.e. the slope of the curve) of the singular values is very high in the beginning and it slows down very fast. The whole curve basically looks like the plot of inverse proportional function. There is a very obvious inflection point on the curve at around the 100th singular values. After that, the decreasing rate becomes relatively slow. In order to maintain as much information as we could, we choose 400 to be the upper limit for the dimension used in K-means algorithm. To find the best result of the algorithm, we try to tune the parameters used in the truncatedSVD step. By sweeping the dimensions from as low as 1 to as high as 400 that we have guessed, we found the best results occurs when the dimension is 165. Next, by sweeping through the number of iterations in the SVD process, we found that 13 is a reasonable number to use. After putting all the best parameters into the algorithm, we have the results as below:

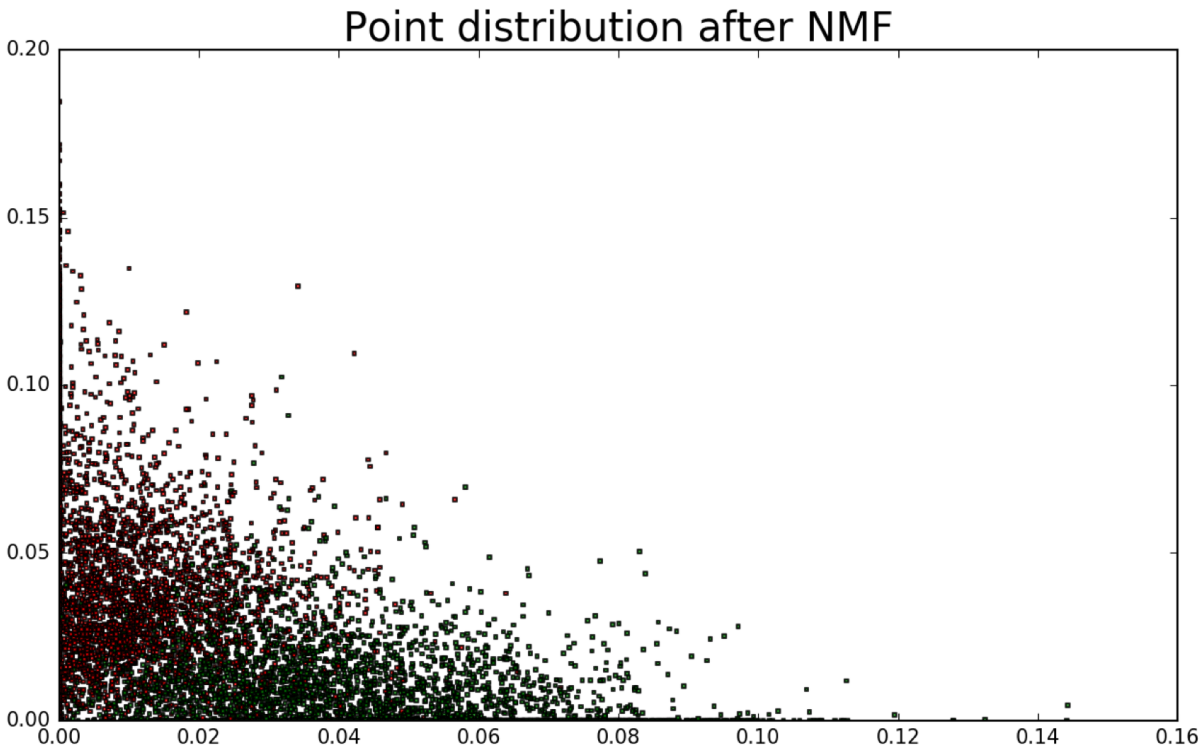
```
Performing truncatedSVD...
Dimension reduction method: truncatedSVD
Homogeneity: 0.545953531329
Completeness: 0.547321564502
Adjusted Rand Score: 0.651865628898
Adjusted Mutual Info Score: 0.545911963211

Confusion matrix:
==============
[[ 281 3698]
 [3425  478]]
==============
```

This result shows that the precision of the algorithm is improved, and the confusion matrix looks “more diagonal” than that in previous question, which indicates better results. We have also noticed that the results of the algorithm are very unstable, probably because K-means is easy to terminate the iteration at some local optimal state. So we have to try multiple times to get the best result.

Next, we will be using another different dimension reduction method called NMF. By applying NMF, we are doing a factorization that split the original matrix V into two matrices W and H. These two matrices have the property that they do not contain any

negative elements. Usually, the number of columns of W and the number of rows of H are selected so the product WH will become an approximate matrix to V. Therefore, NMF can be used to reduce the dimension of the matrix.

To get an intuitive visual sense of how NMF works, we try to reduce the matrix to 2-dimension and produced a scattered plot. Different colors represent different classes. The plot is shown below:



We can see that red dots and green dots have reasonable clusters along with minor overlapping area. If the clustering algorithm works well, it should well separate these two clusters.

By tuning the dimension as we have discussed before, we found that 13 dimensions will yield the best results in this scenario, which is shown below:

```
Performing NMF with logarithm transformation...
Dimension reduction method: NMF
Homogeneity: 0.105516259183
Completeness: 0.218341448256
Adjusted Rand Score: 0.0457631781163
Adjusted Mutual Info Score: 0.105434334165

Confusion matrix:
==============
[[   9 3970]
 [ 815 3088]]
==============
```

Comparing with the results of SVD, the performance of NMF is pretty worse. In order to get better precision, we tried to apply normalization and non-linear transformations to the reduced TF-IDF matrix.

For the non-linear transformation, we choose to use logarithm transformation. However, we must be very careful with those zeros in the original matrix when performing logarithm transformation. The logarithm of 0 is undefined, but we cannot just simply ignore the zeros in the original matrix. This is because log(1)=0, and if we ignore those zeros, then all 0s and 1s will become 0s. This will severely distract the information stored in the original matrix. The way we solve this is by replacing the original zeros to some positive value so that they can be processed by the logarithm operation while the information in the matrix is kept. By sweeping through the replaced value, 0.0008318 turns out to yield the best results, which is shown below:

```
Performing NMF with logarithm transformation...
Dimension reduction method: NMF
Homogeneity: 0.465939264648
Completeness: 0.469125917815
Adjusted Rand Score: 0.565207811578
Adjusted Mutual Info Score: 0.46589037112

Confusion matrix:
==============
[[ 316 3663]
 [3241  662]]
==============
```

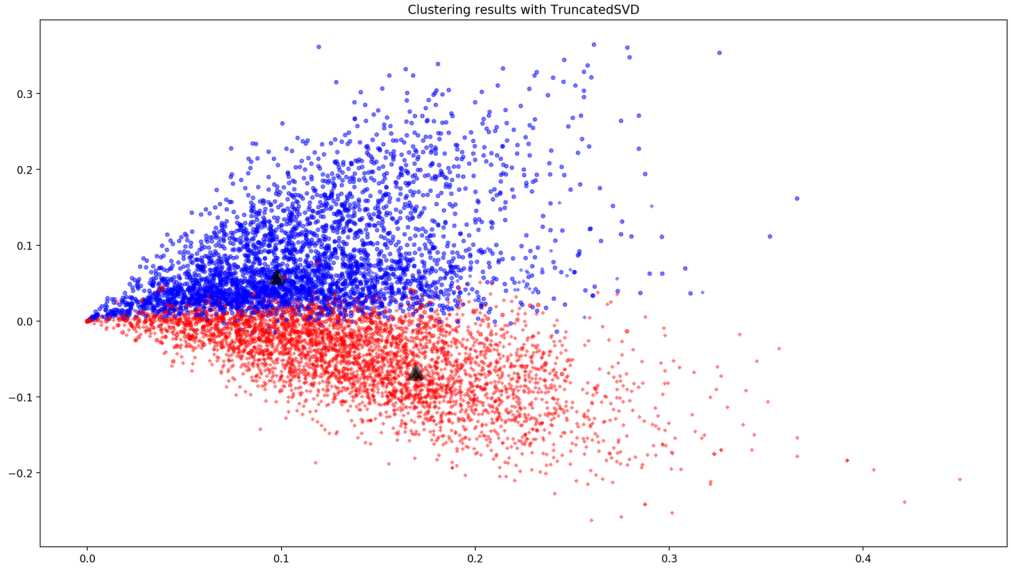
As we can see from the result, the performance is significantly improved by applying the non-linear transformation. Another advantage to use NMF with non-linear transformation is that the results become very stable. That is saying that no matter how many times we

run the algorithm for the same dataset under same parameters, the results are basically the same.

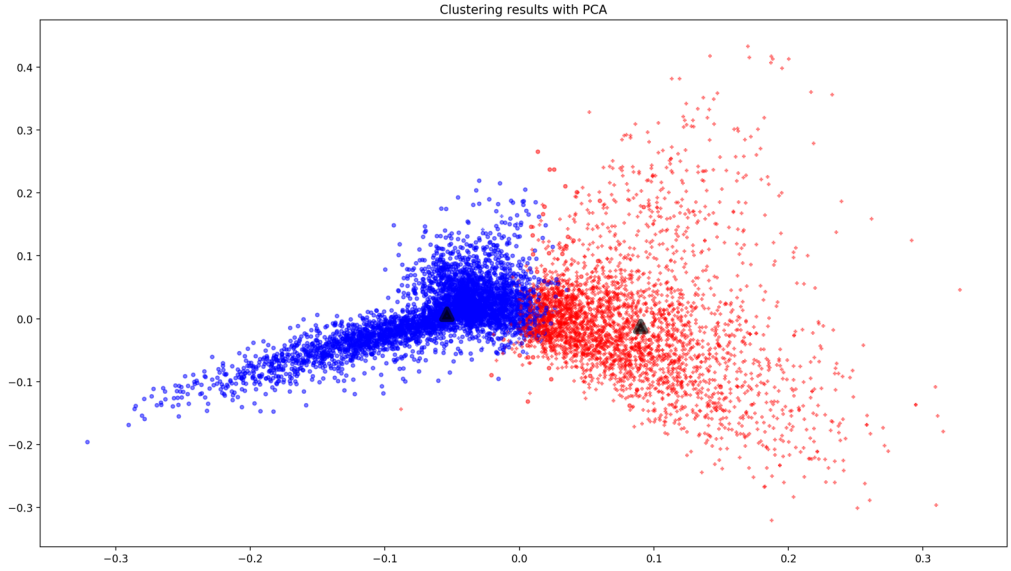
The reason why logarithm transformation behaves outstandingly is, theoretically, it transforms the unknown type of problem into what we have already known so that the data better corresponds to our hypothesis. In this project, we have seen the point distribution of all the data points. If we take the logarithm of every data point, we will better split the two data clusters so that the K-means have a higher chance of correctly cluster those points. This also explains why the NMF behaves more stable than other algorithms.

Question(4)

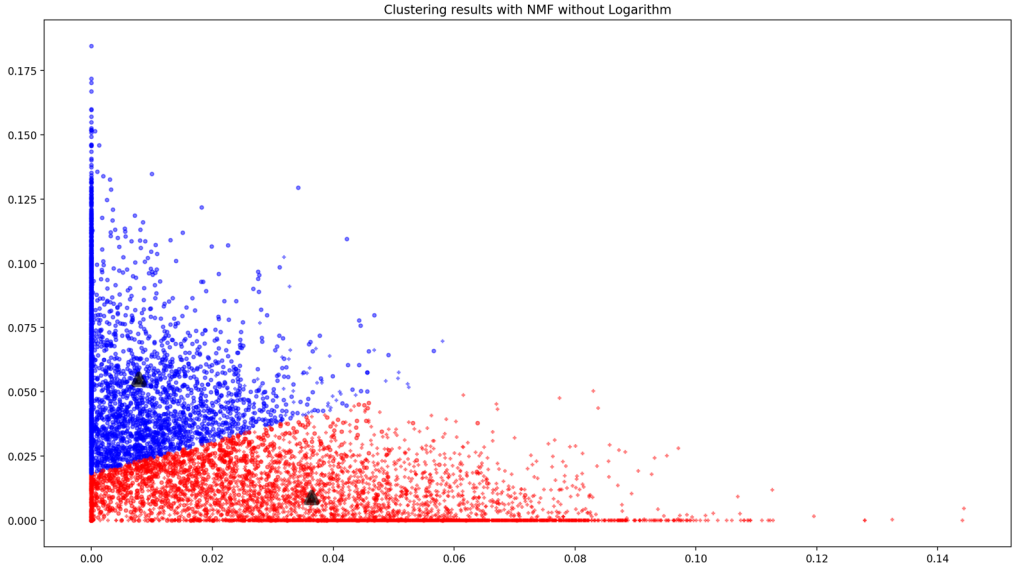
Based on the optimization procedure we did in the last part, we set the reduction dimension values according to the optimized value we got which are: 165 for TruncatedSVD, 3 for PCA, 2 for NMF(Non logarithm) and 9 for NMF(with logarithm). In order to make the clustering results more easy to understand, we can plot the data points with different color and shape which stand for different ground-truth label and the algorithm-clustered label. But the problem is for the optimized case before, the dimension of the final tf-idf matrix are larger than 3, while we know that we can only plot nodes up to 3 dimensions. So a good solution to visualize the clustering results is that we use the optimized dimension reduction to process the tf-idf matrix and do the K-means cluster, than we re-reduced the matrix to 2-dimension to do the plot. Here are the results we got:



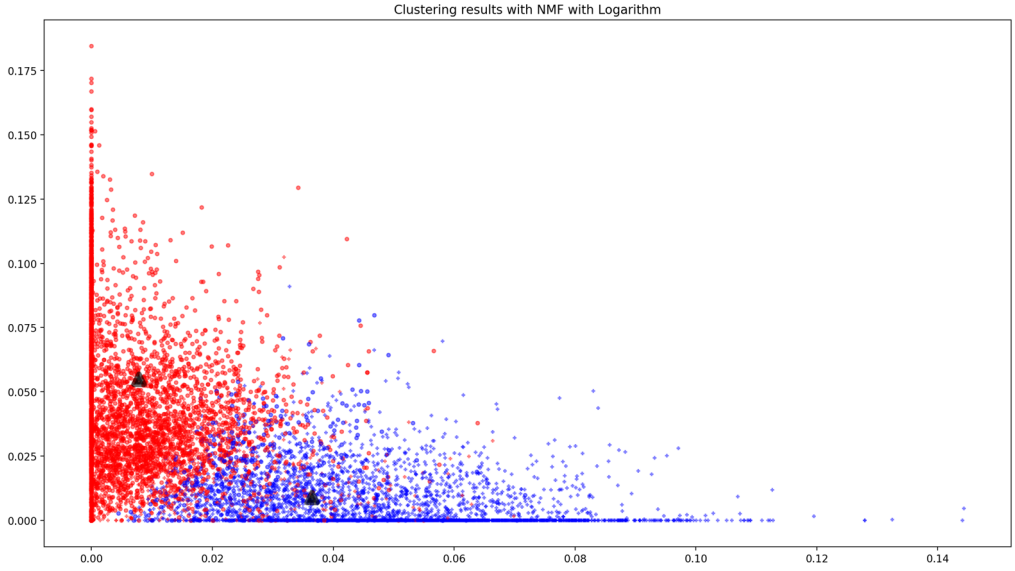
Visualized Clustering Results with TruncatedSVD



Visualized Clustering Results with PCA



Visualized Clustering Results with NMF(no logarithm)



Visualized Clustering Results with NMF(logarithm)

In the visualization above, we marked the data points as circle and plus separately as for the different ground-truth label, and red and blue separately as for the different algorithm-

clustered label. We can see in the results of TruncatedSVD, that the data points distributed almost in a shape of triangle and the ground truth labels are approximately separated along the left angle bisector, where the labels of data points above and below are likely different. And from the color we can also see that, except for some minority of false clustering near the bisector, the K-means algorithm separated the data points along the bisector as well, which is consist with the confusion matrix we inspected before.

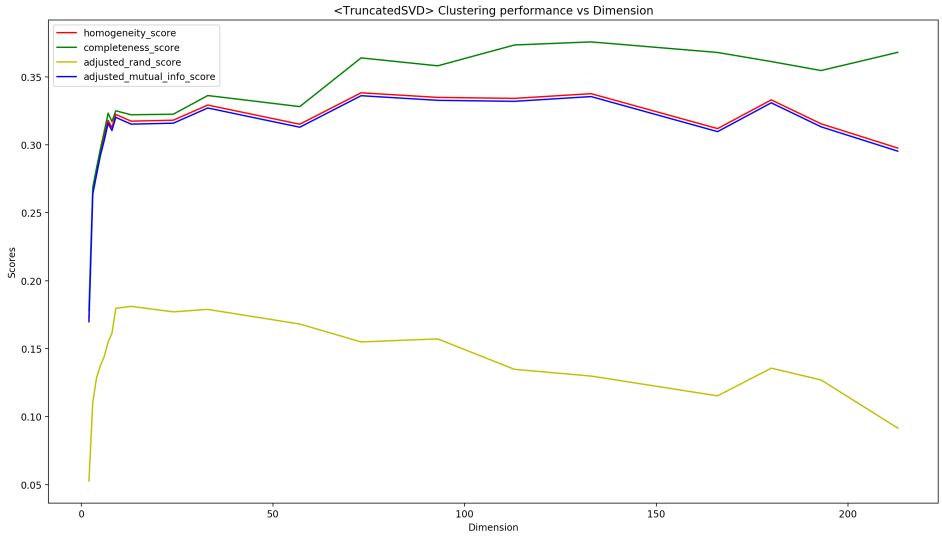
As for the PCA reduction, the data points distributed in a shape like a arrow-head, while the K-means algorithm separated them along the top bisector which is also approximately the partition of the ground truth.

The NMF reduction is more interesting as we can see from the revisualization results above. Firstly, as we analyzed in last part, the data points mostly distributed beneath the (1/x) curve in the plot, while the ground truth labels are approximately separated along the bisector of the 90-degree angle with a considerable amount of false. When the NMF reduction is applied without the logarithm, as we expected, the K-means algorithm separated the data points along the bisector almost linearly, and therefore came with some false clustering points near the bisector. While after we applied the logarithm to the reduced matrix, we can see from the fourth plot, that the partition is no longer absolutely linear, and some false-clustered data points before is now corrected. That is because the logarithm re-distributed the data points, and suppress the error of algorithm arise from the linear partition. And the visualized clustering results are in accord with the clustering scores and confusion matrices we got in question 3 pretty well.

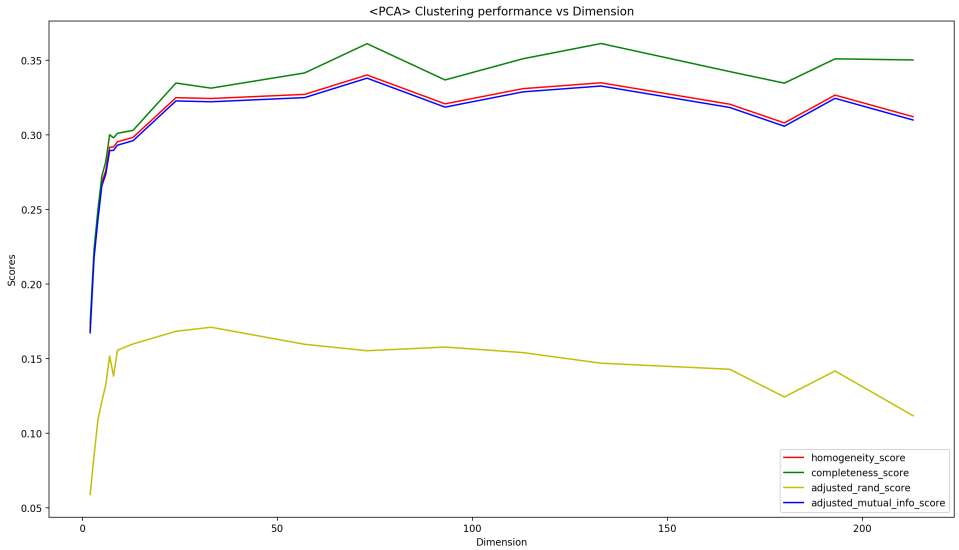
Question (5)

In this part, instead of using the K-means clustering algorithm to cluster the documents into 2 classes, we tried to cluster all the documents from the 20 topics and clustered them into the 20 classes as the ground-truth topics. And also to simplify the process of optimization, we plotted all the four performance scores verse the number of dimension

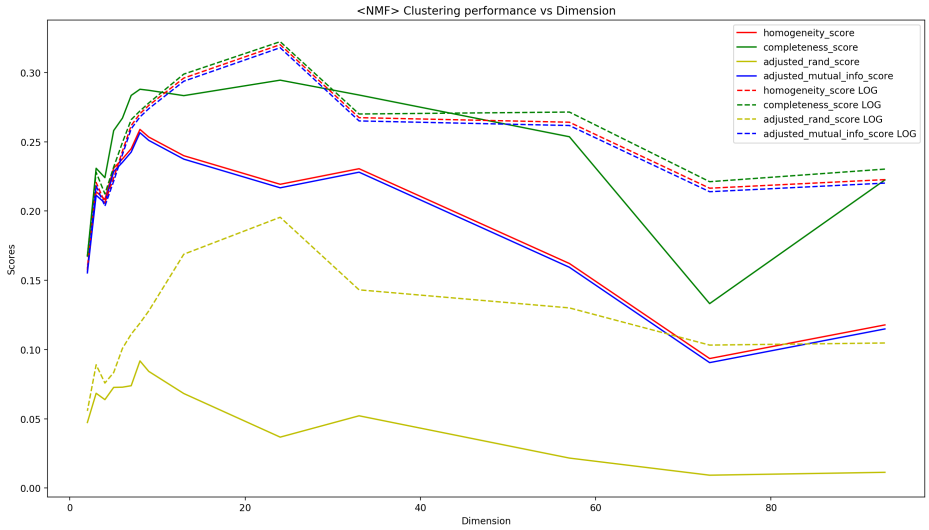
reduced to. And due to the high volume of the confusion matrices in this case, we just simply optimize our dimension reduction based on the performance scores. Here are the results we got for the first run of the dimensions range in *[213, 193, 180, 166, 133, 113, 93, 73, 57, 33, 24, 13, 9, 8, 7, 6, 5, 4, 3, 2]*, and also due to the high volume of calculation in the case of NMF, we truncate the sweep range of dimension reduction in the case NMF to *[93, 73, 57, 33, 24, 13, 9, 8, 7, 6, 5, 4, 3, 2]* to fasten our optimization process:



Clustering scores verse the dimension reduced to <TruncatedSVD>

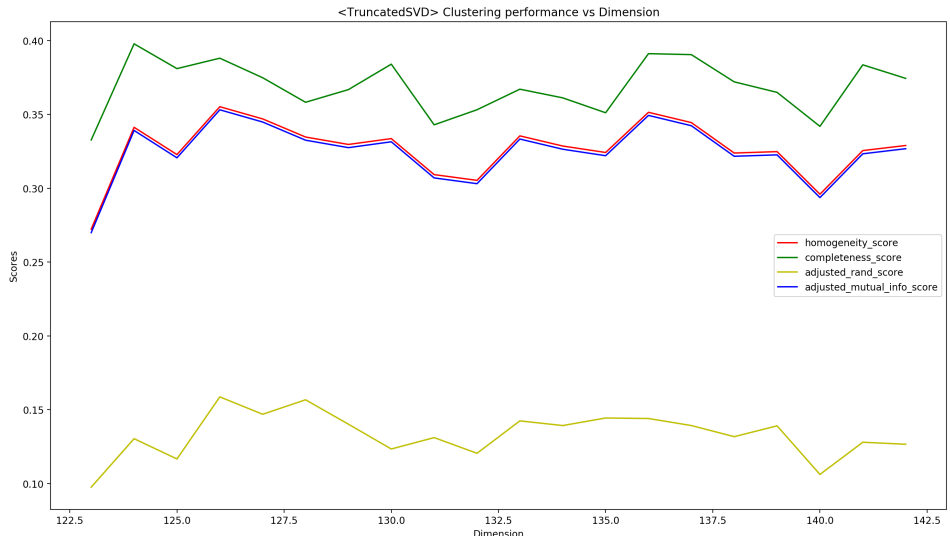


Clustering scores verse the dimension reduced to <PCA>

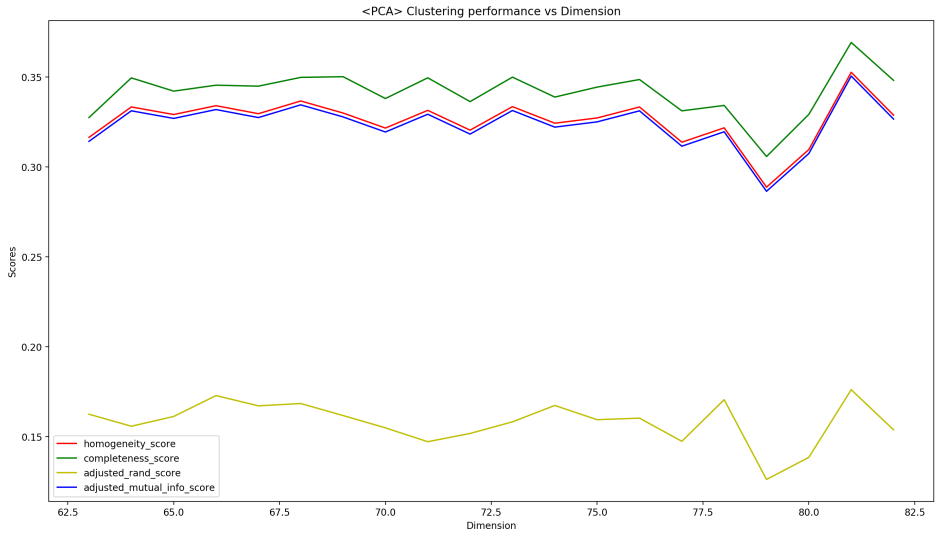


Clustering scores verse the dimension reduced to <NMF>

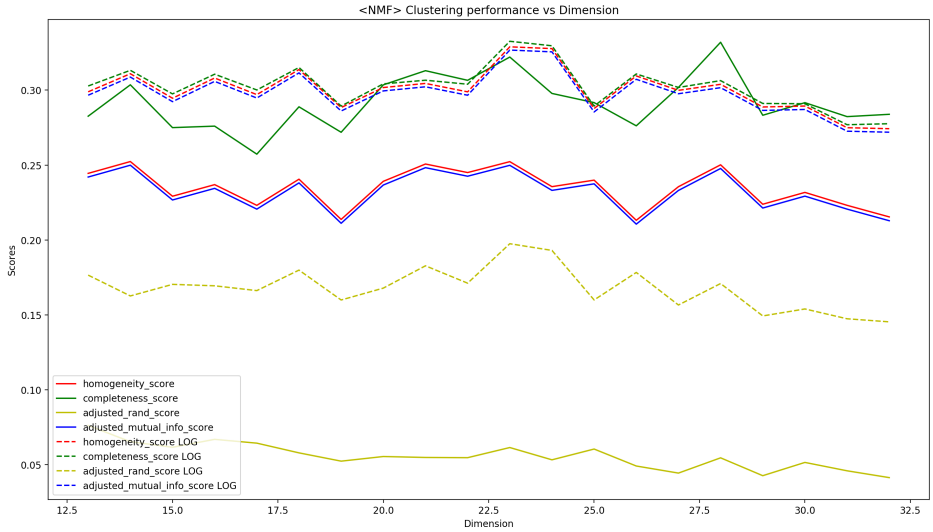
As we can see from the results above, the optimized dimension for three different reduction method fell in different region. For truncatedSVD, the optimized dimension reduced to should be in the range between 113 and 166, while for the PCA method, in the range between 57 and 93, and range between 13 and 33 for NMF(with logarithm transformation). And in order to further optimize the dimension reduction method to the very best level, we tried to do the second round of sweep giving the dimension reduced in the range above. And here are the results:



Second sweep of TruncatedSVD [123, 143]



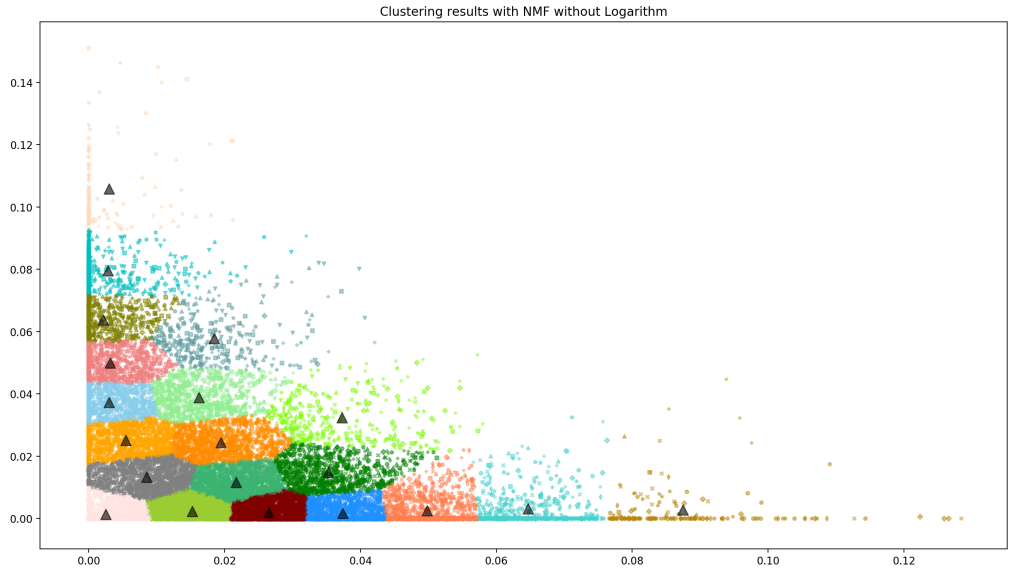
Second sweep of PCA [63, 83]



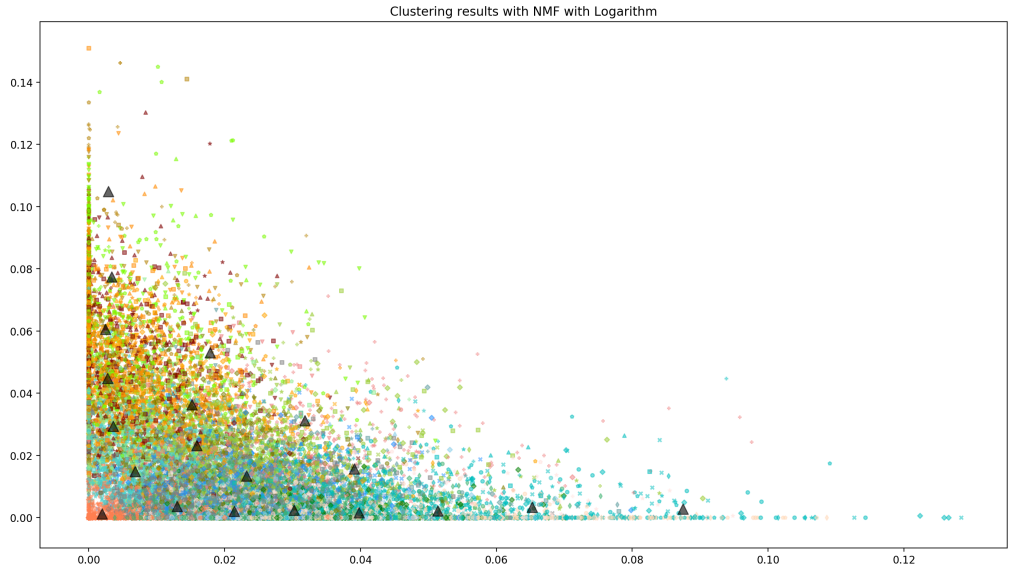
Second sweep of NMF [13, 33]

And now the second sweeping gave us the optimized dimension of reduction which is: 124 for TruncatedSVD, 81 for PCA, 24 for NMF(with logarithm) and 6 for NMF(without logarithm).

Based on those results, we tried to visualize the clustering as we did in the question 4 again. Although the visualized clusters of SVD and PCA are vague due to the large amount of clusters in this case, we can still figure out something from the visualized clustering results with NMF reduction with and without the logarithm transformation:



<NMF without logarithm> Clustering results

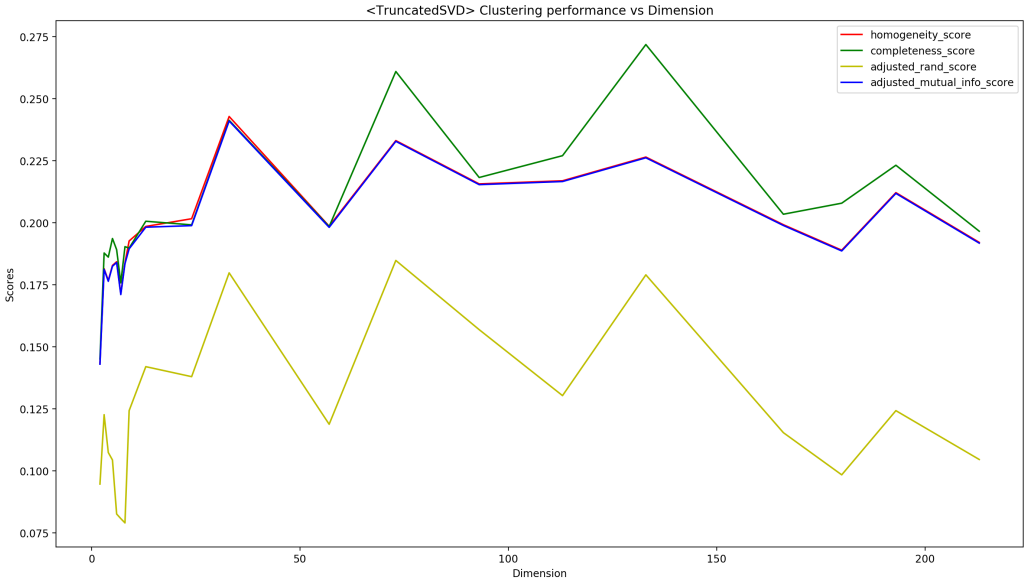


<NMF with logarithm> Clustering results

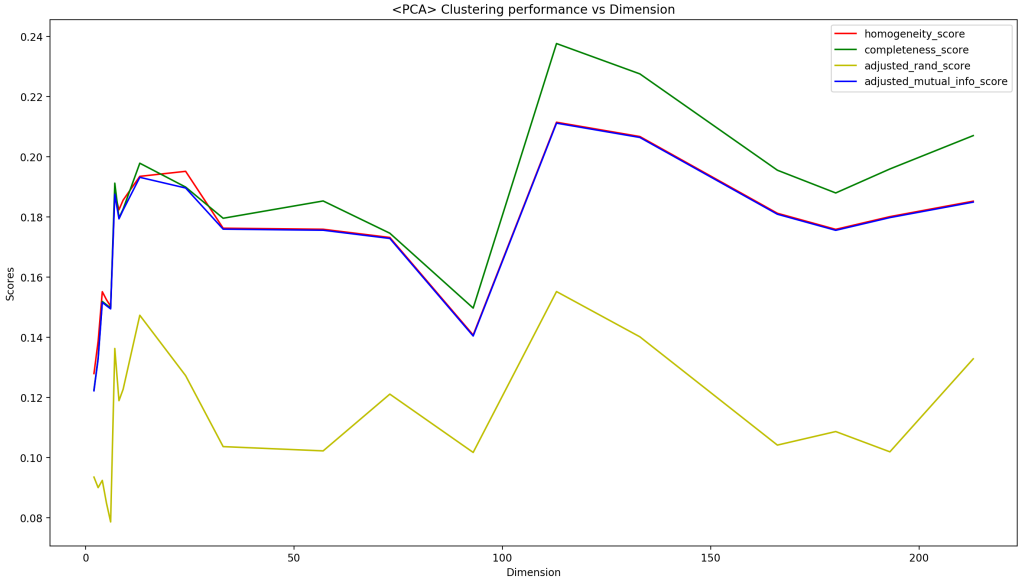
We can see that the K-means clustering algorithm did the almost same partition as we saw in question 4. When we apply the clustering on the NMF(without logarithm) reduced dataset, the partitions between different clusters are almost linear, while for the logarithm case, the partitions are not simply linear anymore and the results are also more accurate.

Question 6

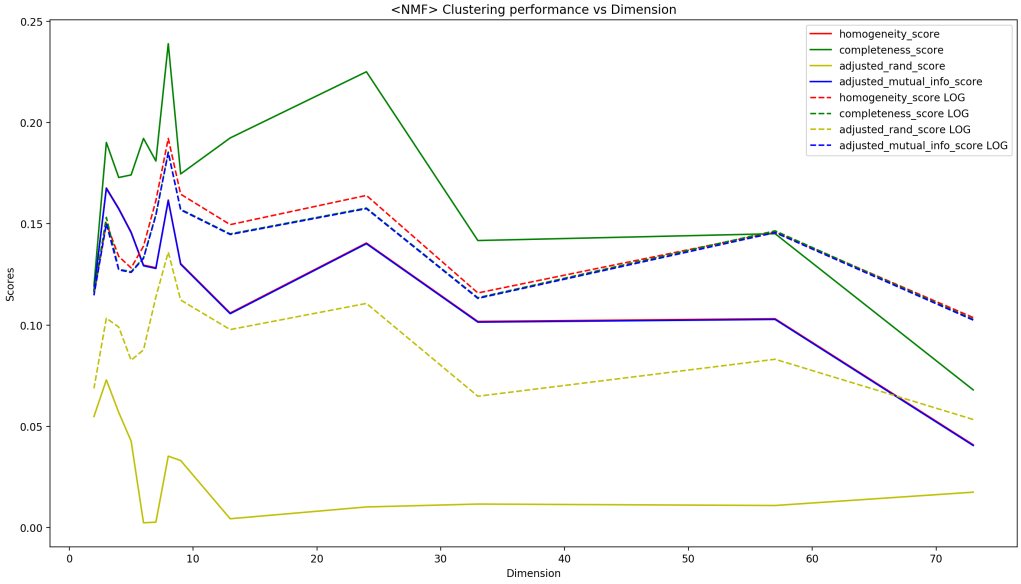
In this final part of project, we did the same dimension reduction then K-means clustering procedure to classify the documents from 20 topics to 6 cluster based on the classes they belong to. For evaluating the performance of the clustering in retrieving the topic-wise classes, we enlarge the dataset to six clusters and twenty sub-classes. We named cluster from cluster0, cluster1, cluster2, to cluster5 representing Class 1, Class 2, Class 3, Class 4, Class 5, and Class 6. Meanwhile, we assigned label 0 to label 5 to those all the documents. And applied different dimension reduction methods on the dataset with K-means clustering coming afterwards. Here are the results we got:



Clustering scores verse dimension reduced to <Truncated SVD>



Clustering scores verse dimension reduced to <PCA>



Clustering scores verse dimension reduced to <NMF>

We can see from the results above, that the optimized reduced dimensions for the three methods we tried were different from the results we got from question 5. Here in the 6-cluster case, the optimized dimension fell in the range of (24, 57) for truncated SVD, (93, 133) for PCA and 8 for NMF(with/without logarithm). And due to the instability of the clustering algorithm, the optimized value we got often varied as we run the script. So we

set our optimized dimension of reduction to: 33 for TruncatedSVD, 133 for PCA and 8 for NMF.

Reference:

1. Official website of scikit-learn package

http://scikit-learn.org/stable/auto_examples/text/document_clustering.html#sphx-glr-auto-examples-text-document-clustering-py

2. Wikipedia - K-means Clustering

https://en.wikipedia.org/wiki/K-means_clustering