EE219 Large Scale Data Mining Models and Algorithms

Winter 2018_Project2

Team member:

Dui Lin (504759948)

Xinyi Jiang (904818856)

Content

1. Introduction **3**

2. Problem Statement and Results **3**

2.1 Question 1 3

2.2 Question 2 4

2.3 Question 3 5

2.4 Question 4 17

2.5 Question 5 21

1. Introduction

Clustering algorithms are unsupervised methods for finding groups of data points that have similar representations in a proper space. Clustering diers from classification in that no a priori labeling (grouping) of the data points is available. In this project, we tried to use unsupervised methods called clustering algorithms for finding groups of data points that have similar representations. Similarly the dataset we used in this project still the “20 Newsgroups”, which is a collection of approximately 20,000 newsgroup documents.

For the purpose of this project, we are going to concentrate on the 8 of the classes which have been shown in the following table.

Computer technology	Recreational activity
comp.graphics	rec.autos
comp.os.ms-windows.misc	rec.motocles
comp.sys.ibm.pc.hardware	rec.sport.baseball
comp.sys.mac.hardware	rec.sport.hockey

Table 1. Subclasses of ‘Computer technology’ and ‘Recreational activity’

2. Problem Statement and Results

2.1 Question 1

Inorder to do the clustering, we need to find a good representation of the data at first. We can just followed the steps in project1, transform the documents into TF-IDF vectors by using min_df=3.

Differ from project 1, stemming function has been removed from the codes. Here are the results we got:

```
In [86]: print (X_tfidf_mindf3.shape)
         print (X_tfidf_mindf3_all.shape)

         (4732, 13477)
         (7882, 19330)
```

we reported the dimensions of the TF-ID matrix for both train set and all the dataset.

2.2 Question 2

In this part, we applied K-means clustering with k=2 using the TF-IDF data. After the preprocessing, we can easily found that the TF-IDF matrix still contains much less valid information. Thus in order to make a concrete comparison of different clustering results, we examine the homogeneity score, the completeness score, the V-measure, the adjusted Rand score and the adjusted mutual info score. Here are the results:

```
Homogeneity:0.417692185876
Completeness:0.451728926094
V-measure:0.434044308255
Adjusted Rand Score:0.435845287754
Adjusted Mutual Info Score:0.417638873884

Confusion matrix:
==============
[[3922   57]
 [1282 2621]]
==============
```

We combine the whole dataset into one and applied the k-means. Different from the previous supervised classification method, we got a much more precise label consistent, which means the clustering performance could be good as long as the larger value in the confusion matrix lay on the same diagonal. All the scores span between 0 and 1; where 1 stands for perfect clustering. And the rand index is similar to accuracy measure, which computes similarity between the clustering results and the ground truth labels, which means that our results seems to be correct.

2.3 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.

So in this part, we firstly would like to determine the effective dimension of the data through inspection of the top singular values of the TF-IDF matrix and see how many of them are significant in reconstructing the matrix with the truncated SVD representation. Specifically, we want to see what ratio of the variance of the original data is retained after the dimensionality reduction.

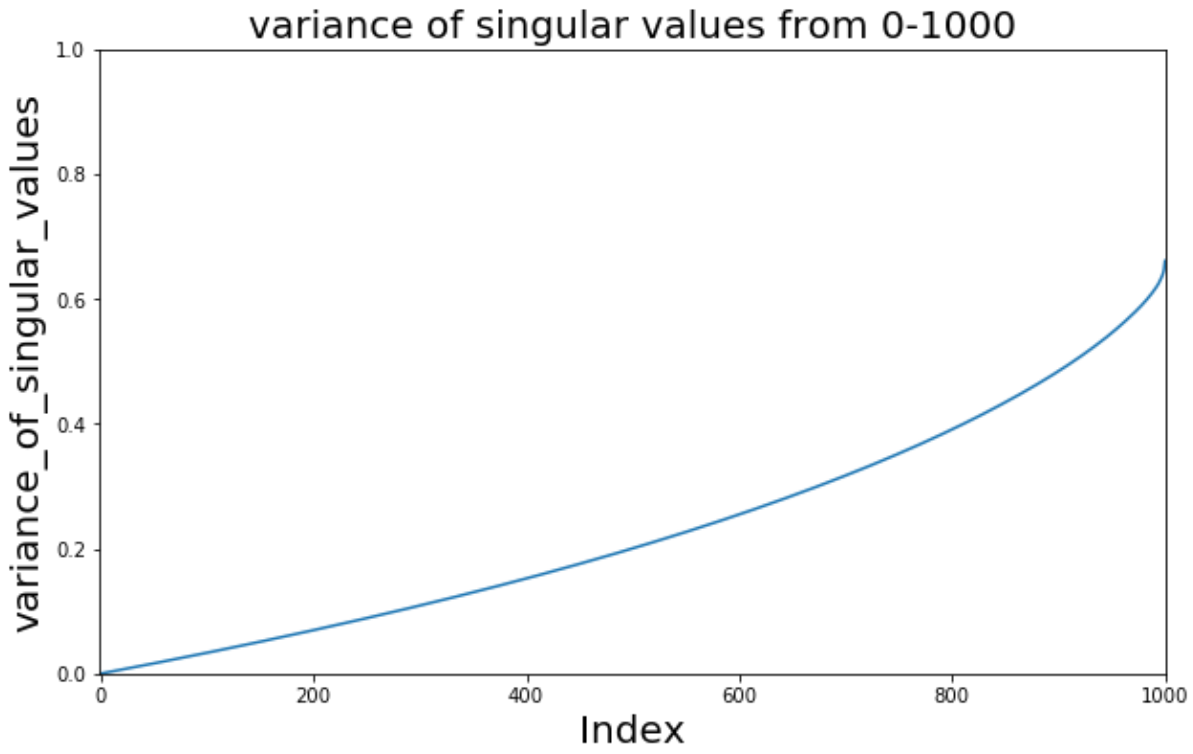

Figure 3.1 Variance of singular values from 0-1000

As the figure 3.1 shows, we divide the accumulated sum of singular values from index 0 to index 1000 by sum of all singular values (more than 7500). We can see that top 1000 singular values will retain around 70% information of the original high dimension matrix, but it could dramatically decrease the dimensions from 7500 to 1000, which is very satisfactory for our data preprocessing.

Then we use LSA, PCA and NMF to reduce dimensions of the data, respectively. We sweep over the number of components from 1 to 300 to find out the best r regarding to the best measure scores. Here are our results:

```
Dimension reduction method:truncatedSVD
Dimension reduction method:truncatedSVD
r = 1
Confusion matrix:
==============
[[3819  160]
 [3781  122]]
==============
--------------------------------------------------------
Performing truncatedSVD...
Dimension reduction method:truncatedSVD
Dimension reduction method:truncatedSVD
r = 2
Confusion matrix:
==============
[[3498  481]
 [ 401 3502]]
==============
--------------------------------------------------------
Performing truncatedSVD...
Dimension reduction method:truncatedSVD
Dimension reduction method:truncatedSVD
r = 3
Confusion matrix:
==============
[[ 490 3489]
 [3506  397]]
==============
--------------------------------------------------------
Performing truncatedSVD...
Dimension reduction method:truncatedSVD
Dimension reduction method:truncatedSVD
r = 5
Confusion matrix:
==============
[[ 448 3531]
 [3454  449]]
==============
```

```
Dimension reduction method:truncatedSVD
Dimension reduction method:truncatedSVD
r = 10
Confusion matrix:
==============
[[3432  547]
 [ 371 3532]]
==============
--------------------------------------------------
Performing truncatedSVD...
Dimension reduction method:truncatedSVD
Dimension reduction method:truncatedSVD
r = 20
Confusion matrix:
==============
[[ 618 3361]
 [3556  347]]
==============
--------------------------------------------------
Performing truncatedSVD...
Dimension reduction method:truncatedSVD
Dimension reduction method:truncatedSVD
r = 50
Confusion matrix:
==============
[[3320  659]
 [ 284 3619]]
==============
--------------------------------------------------
Performing truncatedSVD...
Dimension reduction method:truncatedSVD
Dimension reduction method:truncatedSVD
r = 100
Confusion matrix:
==============
[[3258  721]
 [ 220 3683]]
==============
```

```
--------------------------------------------------
Performing truncatedSVD...
Dimension reduction method:truncatedSVD
Dimension reduction method:truncatedSVD
r = 300
Confusion matrix:
==============
[[3794  185]
 [ 595 3308]]
==============
```

Figure 3.2 Confusion Matrices for truncated SVD

```
Dimension reduction method:PCA
Dimension reduction method:PCA
r = 1
Confusion matrix:
==============
[[ 190 3789]
 [3133  770]]
==============
--------------------------------------------------------
Performing PCA...
Dimension reduction method:PCA
Dimension reduction method:PCA
r = 2
Confusion matrix:
==============
[[3772  207]
 [ 725 3178]]
==============
--------------------------------------------------------
Performing PCA...
Dimension reduction method:PCA
Dimension reduction method:PCA
r = 3
Confusion matrix:
==============
[[3834  145]
 [ 829 3074]]
==============
--------------------------------------------------------
Performing PCA...
Dimension reduction method:PCA
Dimension reduction method:PCA
r = 5
Confusion matrix:
==============
[[ 158 3821]
 [3151  752]]
==============
```

```
Dimension reduction method:PCA
Dimension reduction method:PCA
r = 10
Confusion matrix:
==============
[[3693  286]
 [ 686 3217]]
==============
--------------------------------------------------
Performing PCA...
Dimension reduction method:PCA
Dimension reduction method:PCA
r = 20
Confusion matrix:
==============
[[1084 2895]
 [3320  583]]
==============
--------------------------------------------------
Performing PCA...
Dimension reduction method:PCA
Dimension reduction method:PCA
r = 50
Confusion matrix:
==============
[[2969 1010]
 [3506  397]]
==============
--------------------------------------------------
Performing PCA...
Dimension reduction method:PCA
Dimension reduction method:PCA
r = 100
Confusion matrix:
==============
[[3514  465]
 [3686  217]]
==============
--------------------------------------------------
```

```
Dimension reduction method:PCA
Dimension reduction method:PCA
r = 300
Confusion matrix:
==============
[[ 197 3782]
 [ 130 3773]]
==============
--------------------------------------------------
```

Figure 3.3 Confusion Matrices for PCA

```
Dimension reduction method:NMF
Dimension reduction method:NMF
r = 1
Confusion matrix:
==============
[[1131 2848]
 [2185 1718]]
==============
------------------------------------------------------
Performing NMF with logarithm transformation...
Dimension reduction method:NMF
Dimension reduction method:NMF
r = 2
Confusion matrix:
==============
[[3829  150]
 [ 829 3074]]
==============
------------------------------------------------------
Performing NMF with logarithm transformation...
Dimension reduction method:NMF
Dimension reduction method:NMF
r = 3
Confusion matrix:
==============
[[3963   16]
 [3539  364]]
==============
------------------------------------------------------
Performing NMF with logarithm transformation...
Dimension reduction method:NMF
Dimension reduction method:NMF
r = 5
Confusion matrix:
==============
[[3964   15]
 [3553  350]]
==============
```

```
Dimension reduction method:NMF
Dimension reduction method:NMF
r = 10
Confusion matrix:
==============
[[2885 1094]
 [3900    3]]
==============
--------------------------------------------------
Performing NMF with logarithm transformation...
Dimension reduction method:NMF
Dimension reduction method:NMF
r = 20
Confusion matrix:
==============
[[3963   16]
 [3564  339]]
==============
--------------------------------------------------
Performing NMF with logarithm transformation...
Dimension reduction method:NMF
Dimension reduction method:NMF
r = 50
Confusion matrix:
==============
[[ 972 3007]
 [  21 3882]]
==============
--------------------------------------------------
Performing NMF with logarithm transformation...
Dimension reduction method:NMF
Dimension reduction method:NMF
r = 100
Confusion matrix:
==============
[[ 303 3676]
 [   2 3901]]
==============
--------------------------------------------------
```

```
Dimension reduction method:NMF
Dimension reduction method:NMF
r = 300
Confusion matrix:
==============
[[ 104 3875]
 [ 114 3789]]
==============
--------------------------------------------------
```

Figure 3.4 Confusion Matrices for NMF

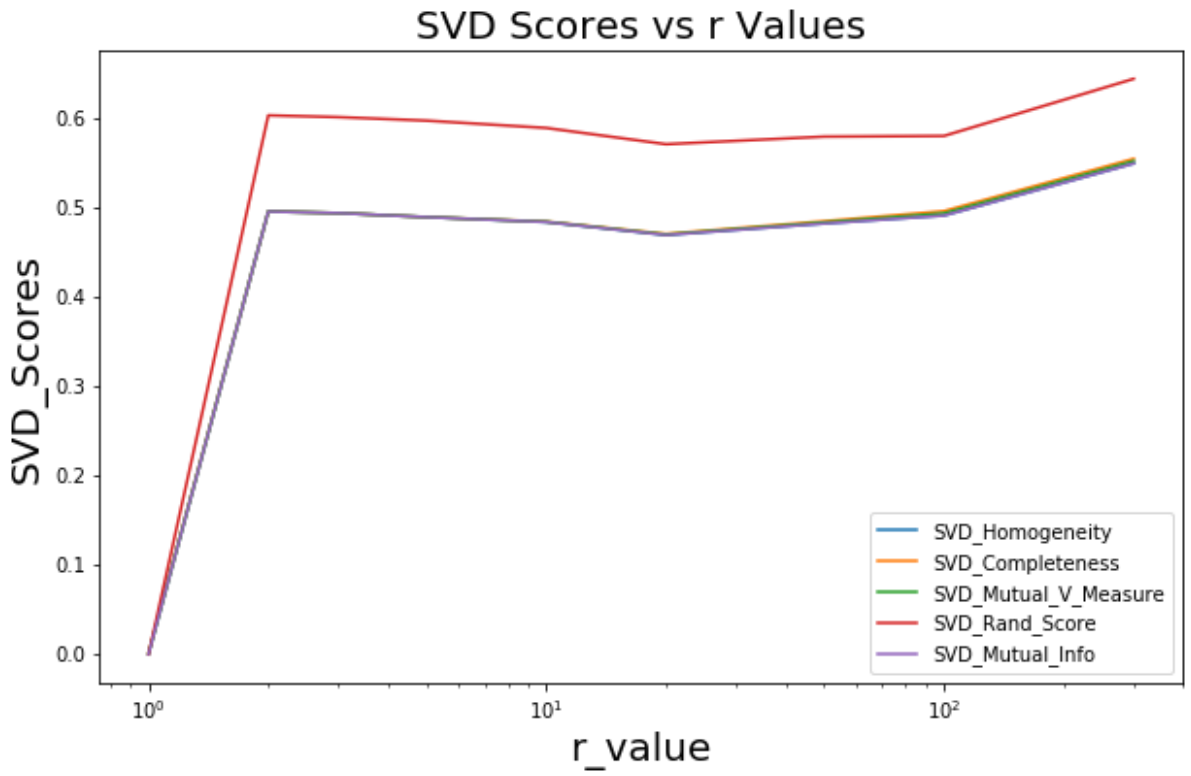

Figure 3.5 SVD Measure Scores v.s. r Values (maximum at 300)

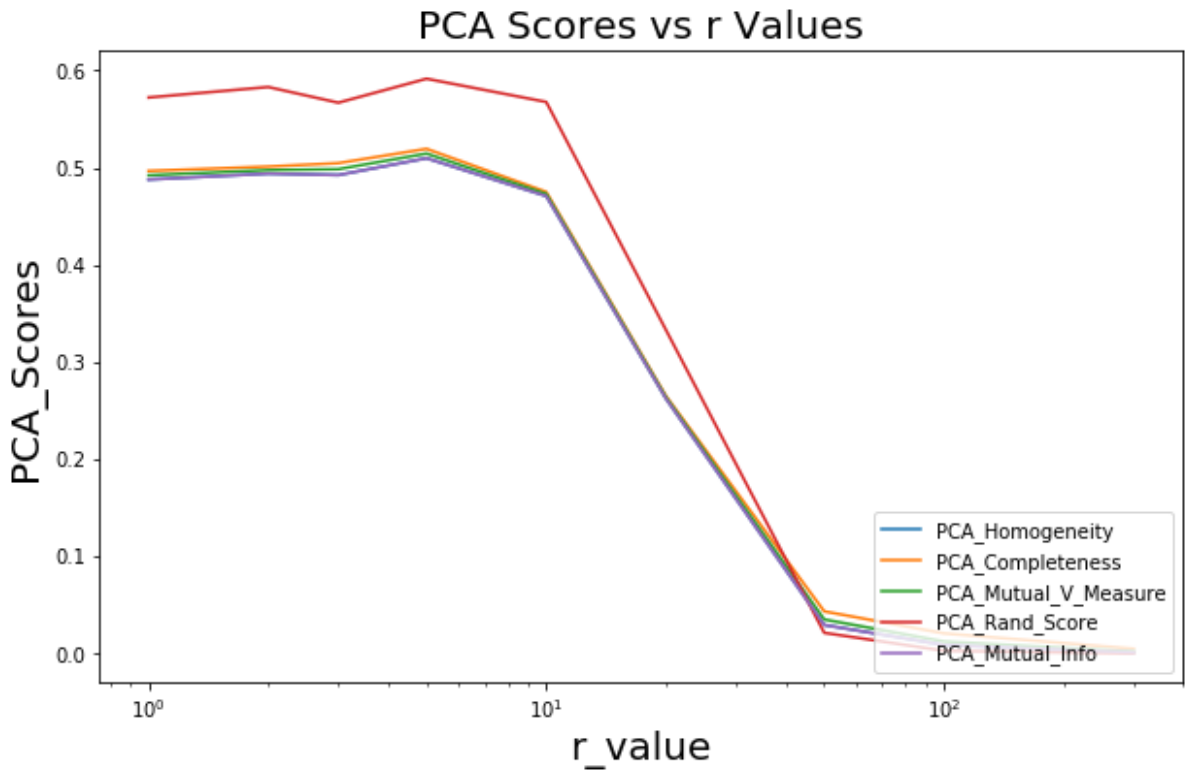

Figure 3.6 PCA Measure Scores v.s. r Values (maximum at 5)

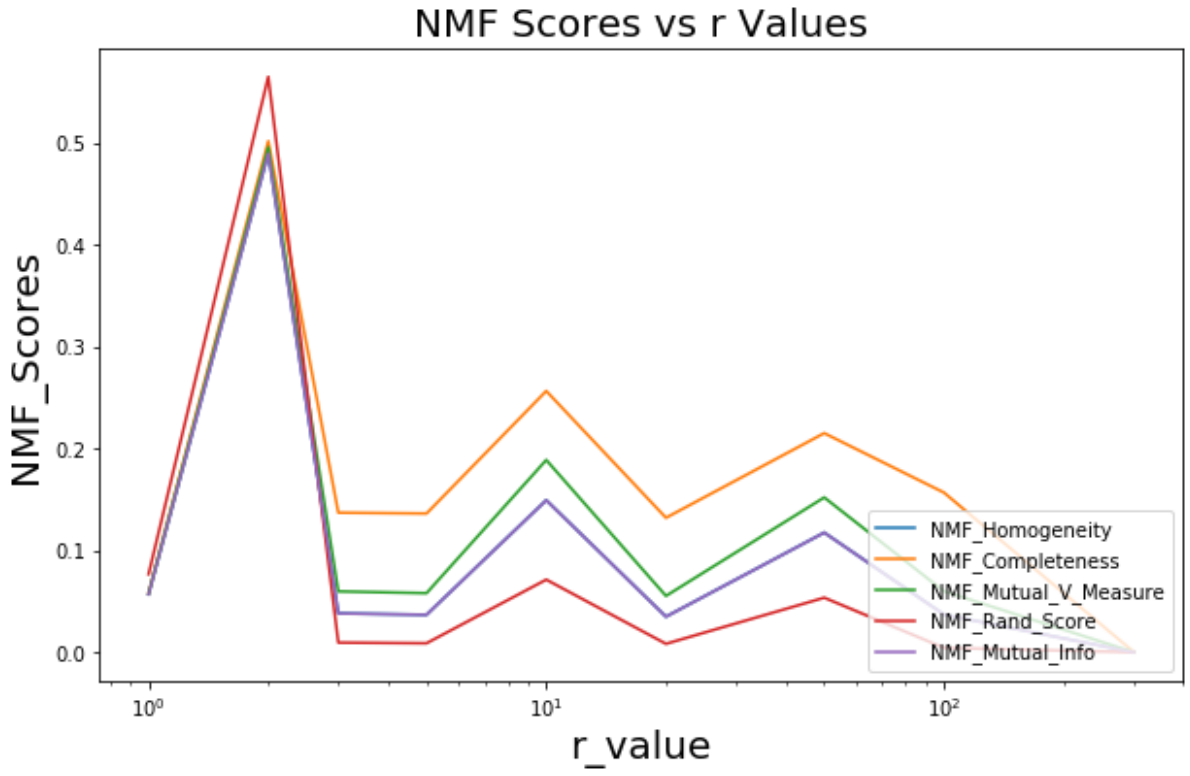

Figure 3.7 NMF Measure Scores v.s. r Values (maximum at 2)

From the figures 3.5-3.7, we have the maximum scores at r = 300, r = 5, r = 2 for SVD, PCA and NMF, respectively. And we can also observe that the curves of scores are not monotonic, which will achieve maximum in the middle of the range of r. That is because for a lower dimension, the retained information is relatively insufficient, on the contrary, for a higher dimension, the matrix will be too sparse for data preprocess and more irrelevant terms will get involved.

2.4 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 before which are: 300 for Truncated-SVD, 5 for PCA, 2 for NMF(Non 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:

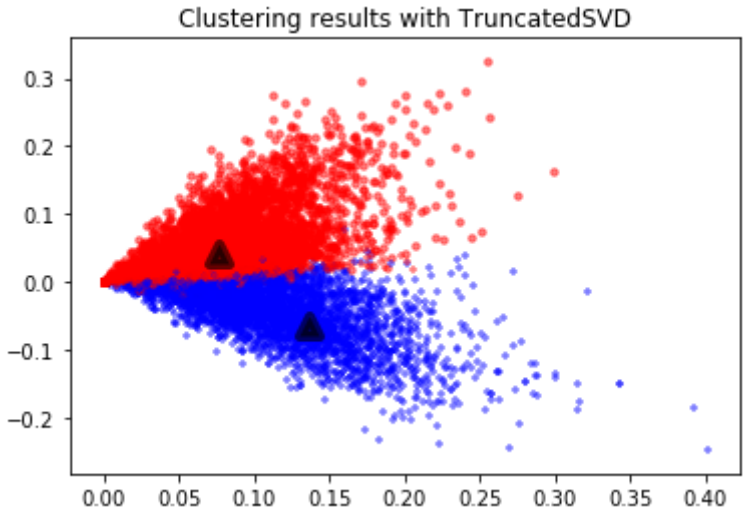

Figure 4.1 SVD Clustering results with normalization applied

```
Performing truncatedSVD with normalization...
Dimension reduction method:TruncatedSVD
Confusion matrix:
==============
[[3768  211]
 [ 561 3342]]
==============
```

Figure 4.2 SVD Confusion Matrix with normalization applied

Figure 4.3 PCA Clustering results with normalization applied

```
Performing PCA with normalization...
Dimension reduction method:PCA
Confusion matrix:
==============
[[ 158 3821]
 [3151  752]]
==============
------------------------------------------------------
```

Figure 4.4 PCA Confusion Matrix with normalization applied

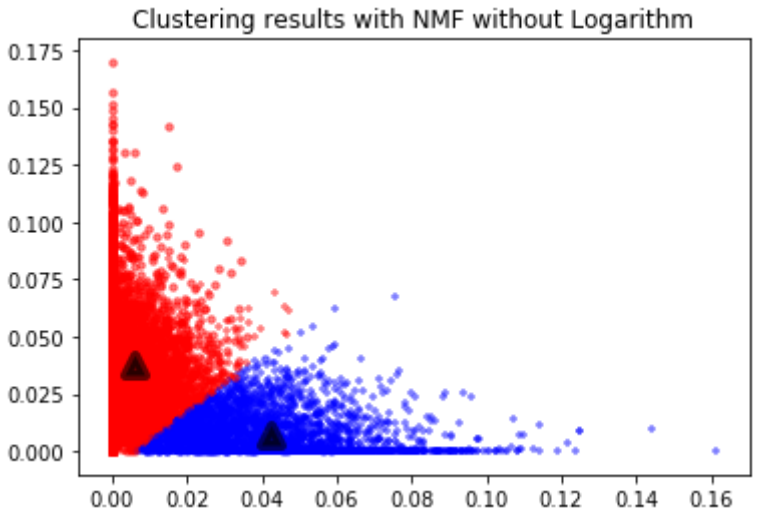

Figure 4.5 NMF Clustering results with normalization but without logarithm transformation

```
Performing NMF without normalization and without Logarithm...
Dimension reduction method:NMF without Logarithm
Confusion matrix:
==============
[[3830  149]
 [ 833 3070]]
==============
```

Figure 4.6 NMF Confusion Matrix with normalization but without logarithm transformation

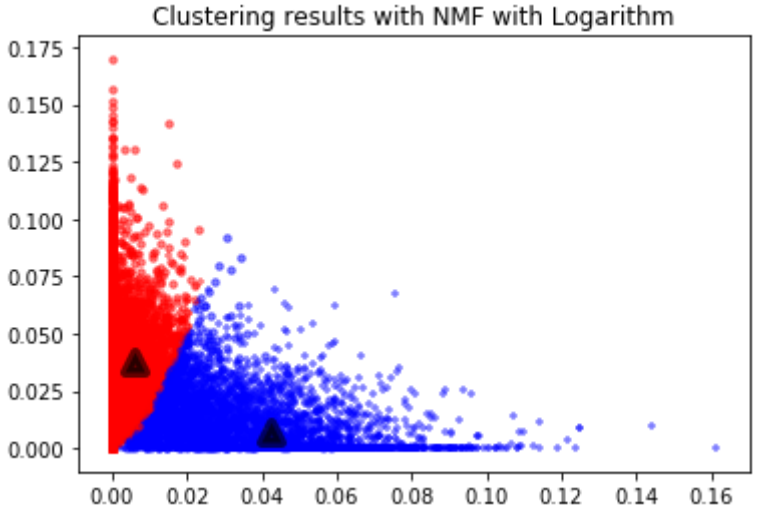

Figure 4.7 NMF Clustering results with normalization and logarithm transformation

```
Performing NMF without normalization and with Logarithm...
Dimension reduction method:NMF with Logarithm
Confusion matrix:
==============
[[3489  490]
 [ 386 3517]]
==============
--------------------------------------------------
```

Figure 4.8 NMF Confusion Matrix with normalization and logarithm transformation

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 Truncated-SVD, 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 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.

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.

2.5 Question 5

In part 5, we want to examine how purely we can retrieve all 20 original sub-class labels with clustering. We need to include all the documents and the corresponding terms in the data matrix and find proper representation through dimensionality reduction of the TF-IDF representation.

We plotted allt the performance scores verse the number of dimension reduced to. Here are the results we got for the first run of the dimensions range in [1,2 ,3, 5, 10, 20, 30, 50].

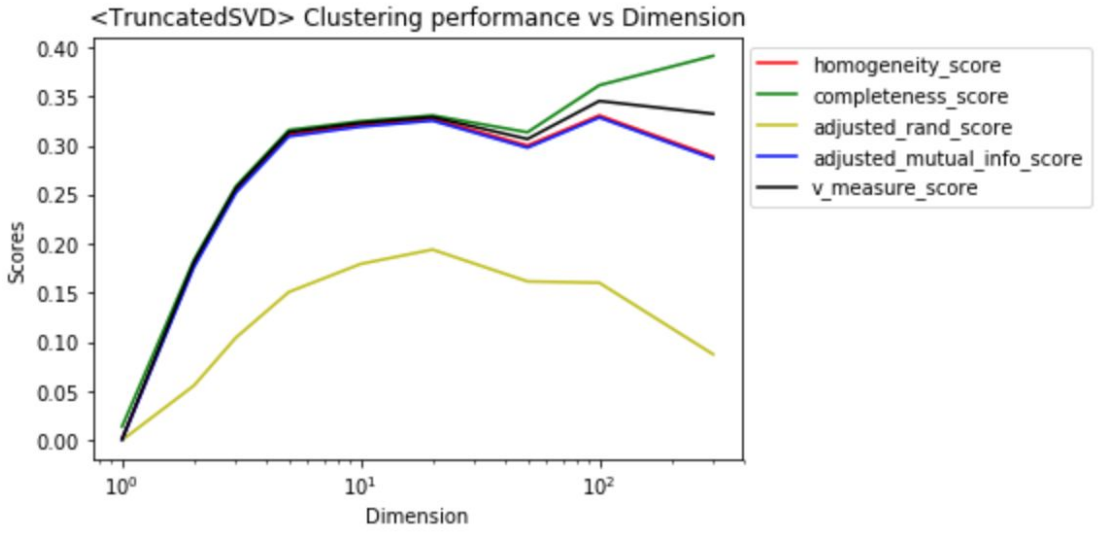

Figure5.1 <TruncatedSVD> Clustering performance vs Dimension

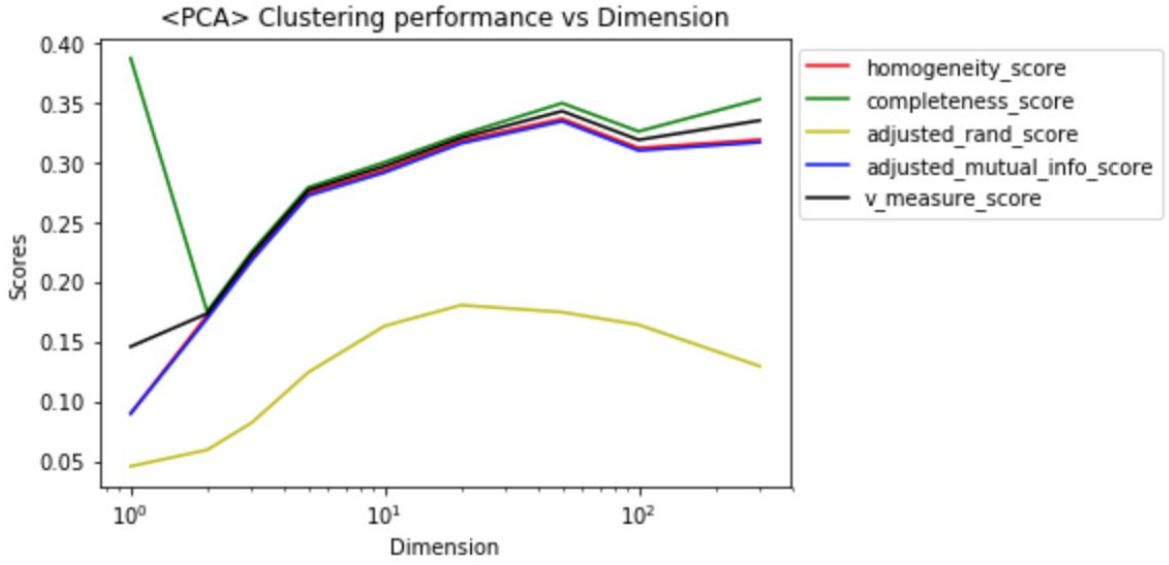

Figure 5.2 <PCA> Clustering performance vs Dimension

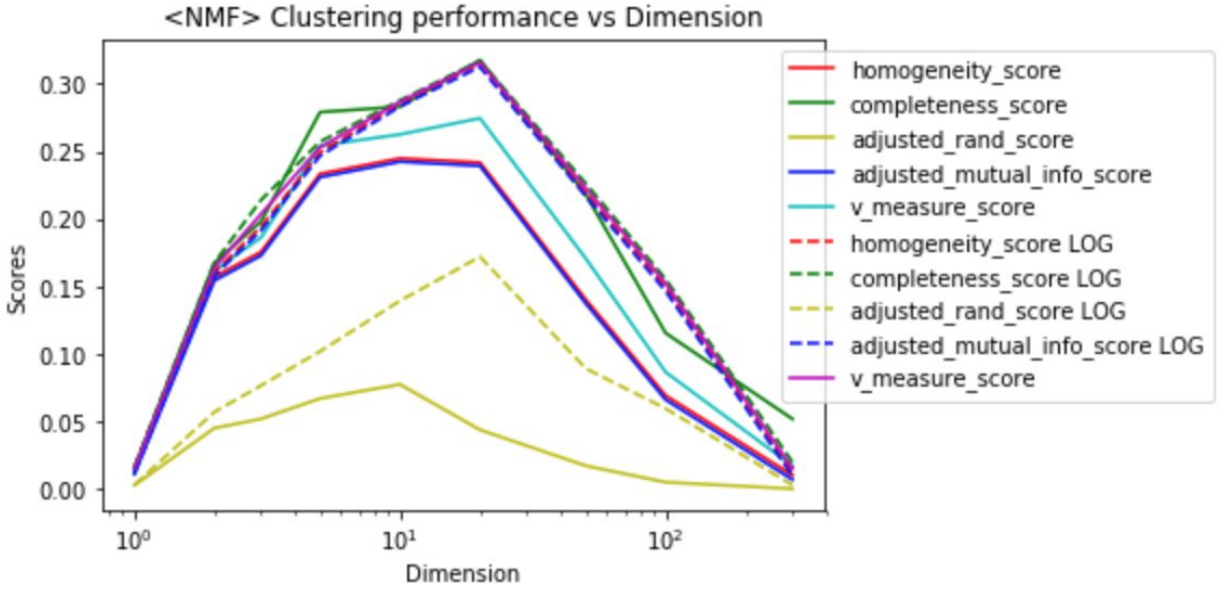

Figure 5.3 <NMF> Clustering performance vs Dimension

According to those results, we tried to visualize the clustering result.

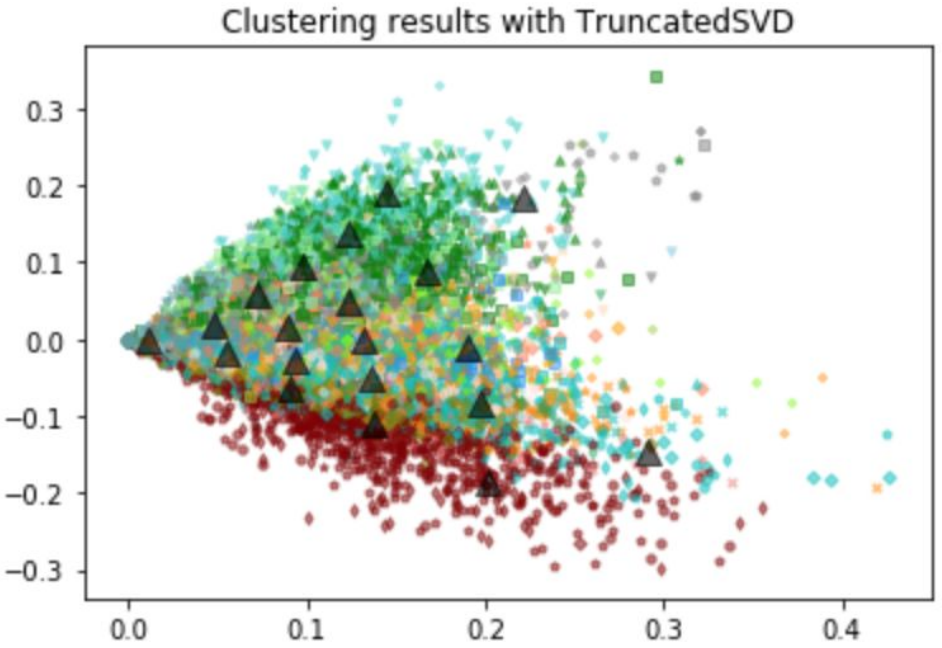

Figure 5.4 Clustering results with TruncatedSVD

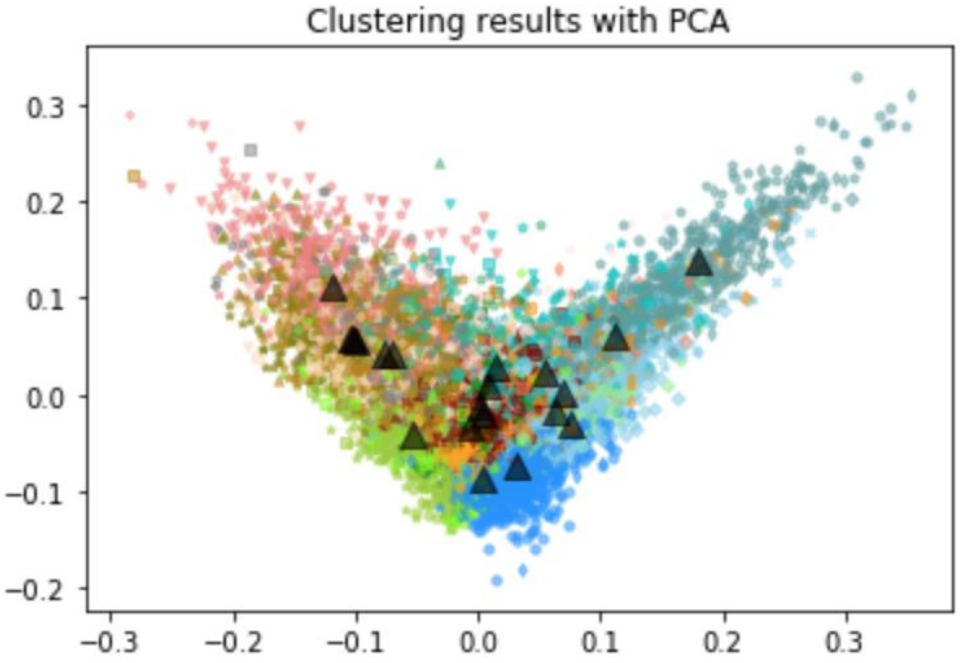

Figure 5.5 Clustering results with PCA

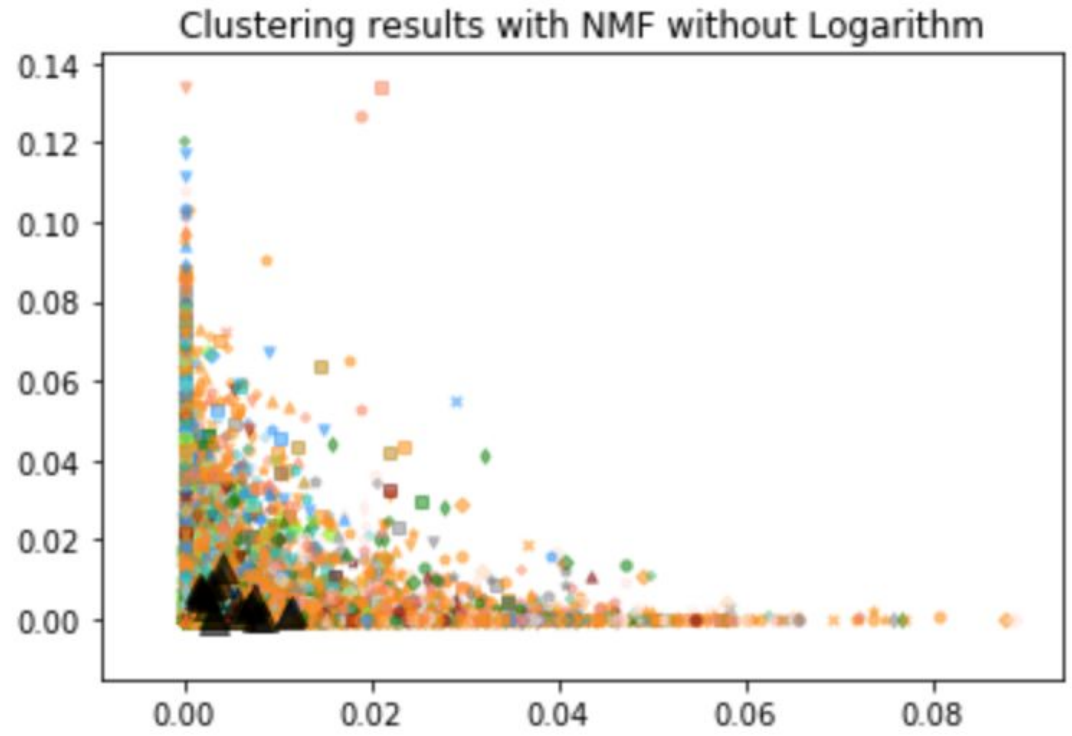

Figure 5.6 Clustering results with NMF without Logarithm

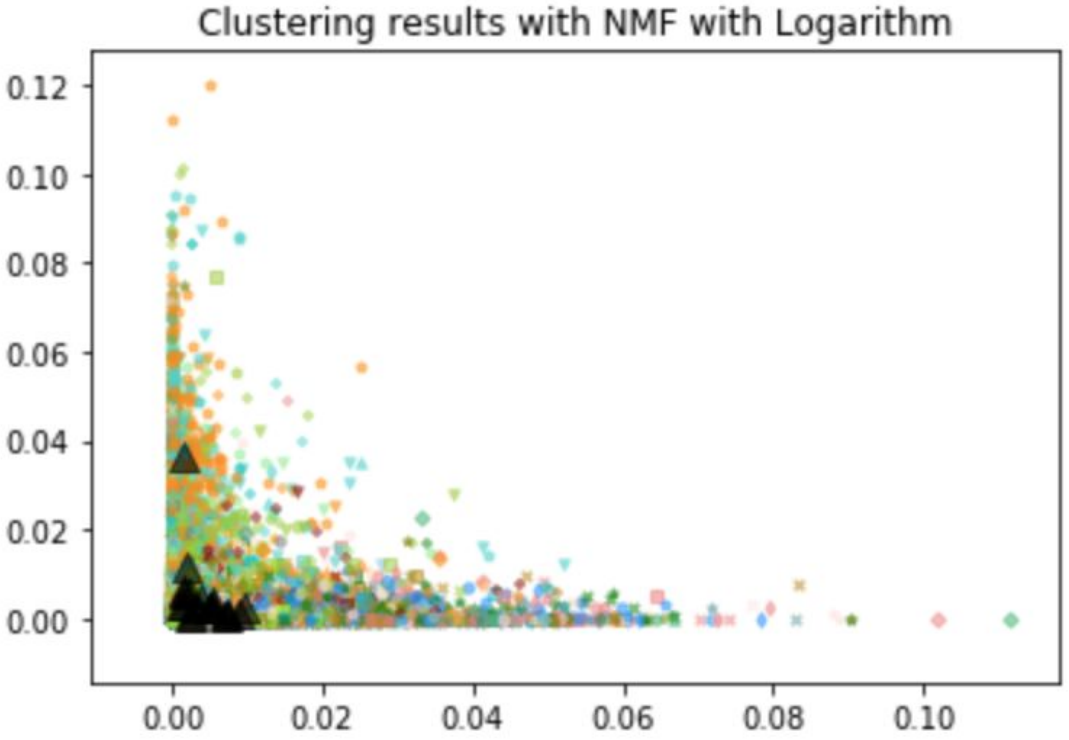

Figure 5.7 Clustering results with NMF with Logarithm

We can see that K-means clustering algorithm did the almost same partition. When we apply the clustering on the MNF 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 accurate.