EE219 Project 4
K Mean Clustering

Winter 2017

Boyang Cai 304330123
Manni Chen 304145309
Zhuoqi Li 004855607

Part A. TFIDF Matrix Construction

Follow the same steps as we did in project, we extracted the TFIDF matrix for the 8 categories in the topic of computer and recreational activity. This matrix has 7882 rows, which stands for documents and 67764 columns as terms. Note that we took out all stem and stop words as we did in project 2.

Part B. Clustering Score Evaluation

Use the TFIDF matrix obtained in part a, we feed it directly into the k-mean clustering algorithm with sklearn kmean library. We set the max iteration number to be 200 and tolerance threshold to be 1e-5. The clustering parameter is set to be 2. In order to evaluate the performance of the k mean clustering algorithm, we used the homogeneity, completeness, adjusted rand and adjusted mutual scores. The result is displayed as follows:

init	Homogeneity	Completeness	ARI	AMI
k-means++	0.421	0.455	0.44	0.421
random	0.416	0.451	0.429	0.415

K-means++ and random stand for two kmean clustering initialization methods we used in which k-means ++ selects initial cluster centers for k-mean clustering in a smart way to speed up convergence.

As revealed from the table for both method, the score is around 0.4 ~ 0.45 as oppose to the perfect score 1, which indicates that the clustering performance is not very good.

To further confirm our evaluation based on these scores, we also drawed the confusion matrix:

Total = 7882	Predicted 0	Predicted 1
Actual 0	1273	3924
Actual 1	2630	55

First look on the confusion matrix you might conclude that the clustering result is really bad, if not horrible. However, if we permute the row, switch the first row with the second row, the confusion matrix now becomes:

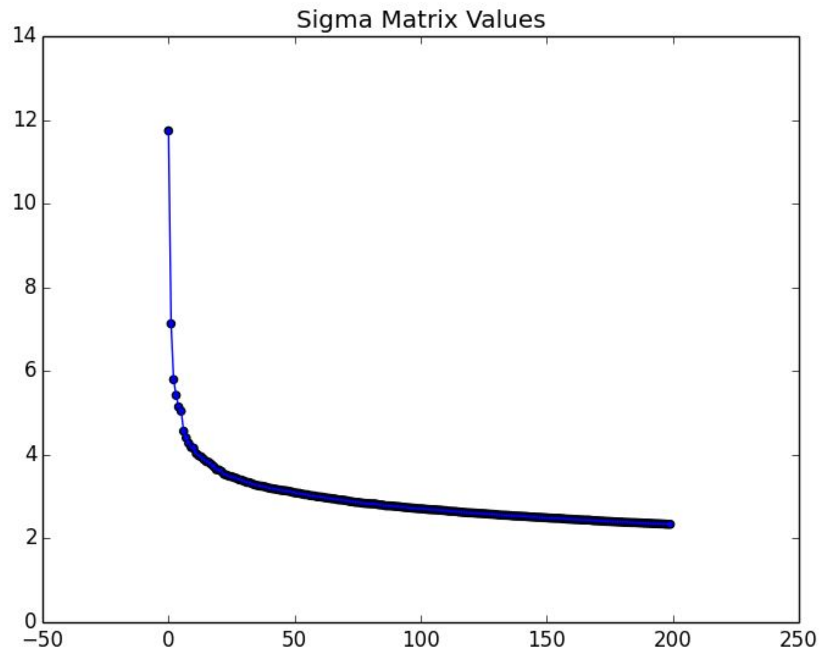
Total = 7882	Predicted 0	Predicted 1
Actual 1 (0)	2630	55
Actual 0 (1)	1273	3924

Now the result looks much better and the matrix now looks more diagonal. Usually we can't do this permutation but for the purpose of this project, it does NOT matter how we label the cluster but only how items in the predicted cluster matches with those in the actual cluster matters. This is also why we use those four scores to evaluate the clustering performance. According to the confusion matrix, the prediction rate is 83.15%.

Part C. Feature Reduction Clustering

As shown in part b, the original matrix has too many dimensions that it confuses the k mean clustering algorithm. Therefore in this part, in order to enhance the clustering performance, instead of directly feeding the original TFIDF matrix, we first perform a PCA or NMF to project the matrix onto a lower dimension, from the 7882 X 67724 matrix to 7882 X K matrix. With less dimension, it makes the algorithm much easier to calculate and compare euclidean distance between data points.

To find this proper k value, we need to first look at the singular values in the sigma matrix of SVD of the original TFIDF matrix. The following diagram display first 200 values in the sigma matrix:



The graph above reveals there is an exponential decaying pattern in sigma values and after the 200th singular value, sigma values stabilize around 2. Therefore, the range of values to be chosen for k is from 0 to 200 as any value larger than k will just yield almost identical result as k = 200.

The way we choose k value in this range is by sweeping through all k values in the range and pick the one that yield the best result. We finally resided on **50** as our k value to perform PCA and NMF analysis. However, there is no significant differences in performance for k in the range of 30 to 50. The following table shows four scores for both PCA and NMF analysis:

Analysis	Homogeneity	Completeness	ARI	AMI
PCA	0.43	0.462	0.452	0.43
NMF	0.472	0.492	0.525	0.472

Compared with clustering scores obtained from last part, scores does increase after PCA and NMF analysis. However, the increase is not significant at all.

To further evaluate how NMF and PCA analysis affect clustering performance, let's check the confusion matrix again:

NMF:

Total = 7882	Predicted 0	Predicted 1
Actual 0	2884	105
Actual 1	1019	3874

PCA:

Total = 7882	Predicted 0	Predicted 1
Actual 0	2670	57
Actual 1	1233	3922

Confusion matrices above shows the prediction rate of NMF and PCA is 85.7% and 83.6%, which is slightly larger than what we obtained in the previous part. However we are not very satisfied with the scores obtained, we also tried normalizing features and non-linear transformation of the data vectors in attempt to further enhance the clustering performance.

N_component = 50

Analysis	Homogeneity	Completeness	ARI	AMI
PCA_norm	0.517	0.520	0.611	0.517
NMF_norm	0.492	0.498	0.581	0.492

Analysis	Homogeneity	Completeness	ARI	AMI
PCA_log	0.414	0.453	0.420	0.414
NMF_log	0.469	0.489	0.521	0.469

The tables above display the clustering scores applied normalization, logarithmic non-linear transformation. As from the results above, both clustering performance are improved after the normalization. For logarithmic transformation, it does improve the clustering performance of the NMF by a little bit but however make it worse for the PCA.

Normalization proves to be an effective technique to enhance the performance for both PCA and NMF. The reason behind this phenomenon is that by doing normalization on the matrix column, we performed a data preprocessing that reveals the relationship exist in between different documents which make it easier for the k-mean cluster algorithm to distinguish them.

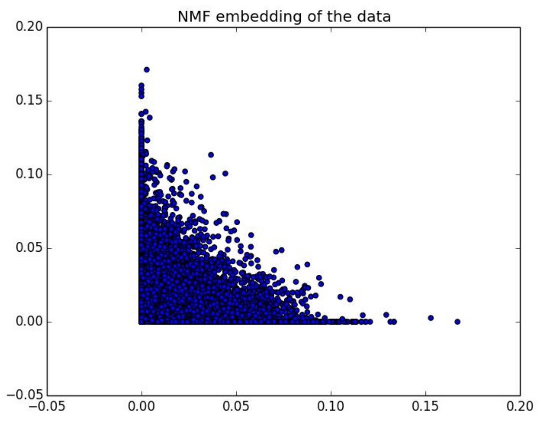
Analysis	Homogeneity	Completeness	ARI	AMI

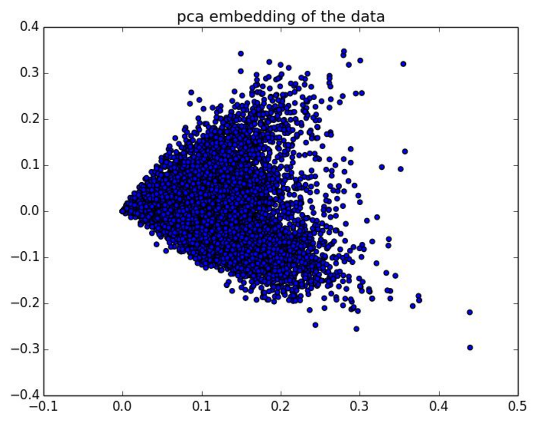
PCA_log w/norm	**0.531**	**0.532**	**0.634**	**0.531**
NMF_log w/norm	0.491	0.496	0.581	0.491

Since both normalization and logarithmic transformation proves to be helpful in enhancing clustering performance, it makes senses that to combine them both. As reveals from the table above, it turns out that our guess is correct. We obtained the best performance score by combining them together. The best purity scores we got comes from the **normalization on logarithmic PCA reduced matrix.**

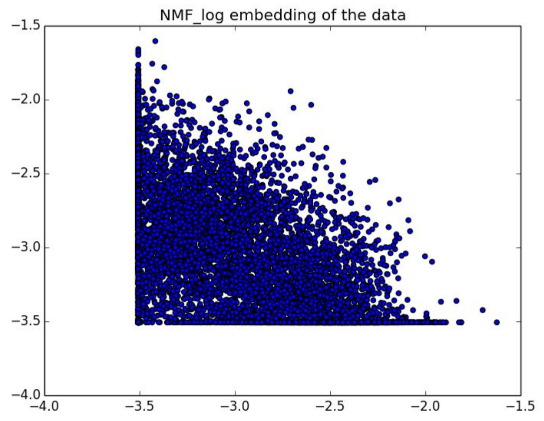
For data visualization, when we applied NMF with ambient parameter at 2 and plotted the resulting points as shown below, we found that points are not evenly distributed in the plot. Lots of points have small values and are located close to the origin. In order to make the make points more evenly distributed for clustering, we applied another non-linear transformation. The non-linear transformation takes both exponential function with the power of 0.4, which allow values smaller than 1 be expanded to large scale, and another log function, which let the transformed value more linearly distributed.

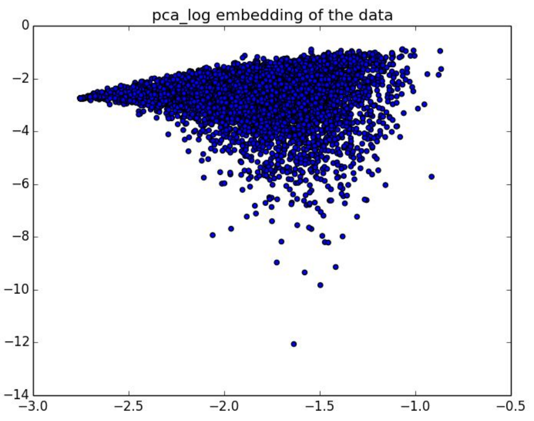
To further justify why logarithmic transformation is good, we choose the k value to be 2 and then plot the resulting data points:





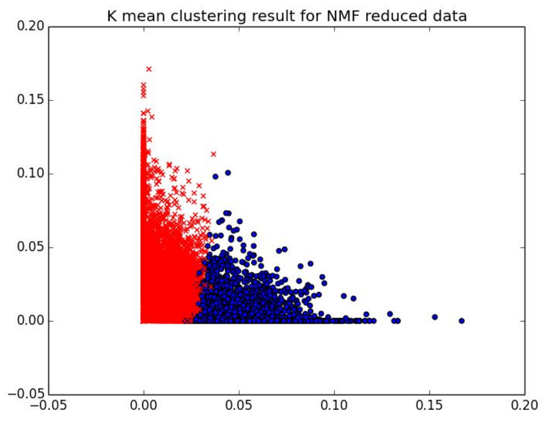
As we can see from two diagrams above, for the NMF embedding of the data, there seems to exist an exponential pattern in data points distribution. Most of the data crowd around the origin giving the k mean clustering algorithm a hard time to distinguish them. However, it is not the case for the PCA embedding.

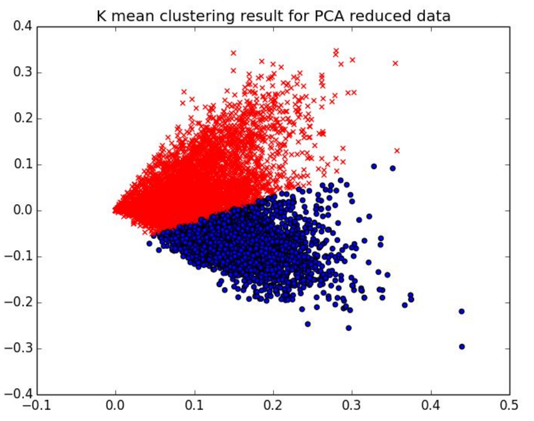




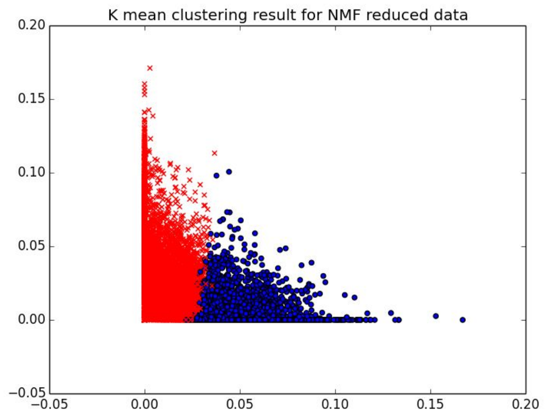
Two graphs above exhibits the effect of logarithmic transformation. Clearly, it helps to scatter data points in the NMF embedding. However, it makes the PCA embedding more crowded. By visualizing the effect of logarithmic transformation, it explains why logarithm help improve the clustering scores on NMF embedding of the data but make it worse for the PCA. It also demonstrate that logarithm is a good candidate for TFIDF data.

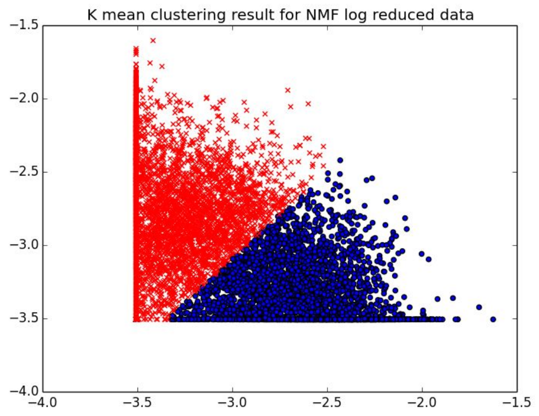
Part D. 2-dimensional Clustering Visualization





These two diagram show the clustering labeling result for both NMF and PCA results.





The two diagrams above display how logarithmic transformation affects the clustering performance of NMF reduced data. Obviously, compared with the diagram on the left, clusters in the diagram on the right is more clear and more evenly distributed. This result matches with the better purity scores obtained for logarithmic transformed data.

Part E. Evaluate Original Sub-class Label Clustering

Follow the same step in part C, we expand our data source to be all 20 sub-class in the database. We found the clustering algorithm yield the best result when n_componenets is equal to 50. Since we have 20 sub_classes and set the kmean max_iteration at 300, it is better to change the initial number of clusters to a greater number for better performance. In this case, we finally choose the initial number of cluster to be 50 which returned us the best result.

N_component = 50, k_init for kmean = 50

Analysis	Homogeneity	Completeness	ARI	AMI
PCA	0.340	0.397	0.108	0.338
NMF	0.305	0.374	0.087	0.303

Analysis	Homogeneity	Completeness	ARI	AMI
PCA_norm	0.389	0.404	0.234	0.387
NMF_norm	0.343	0.359	0.180	0.341

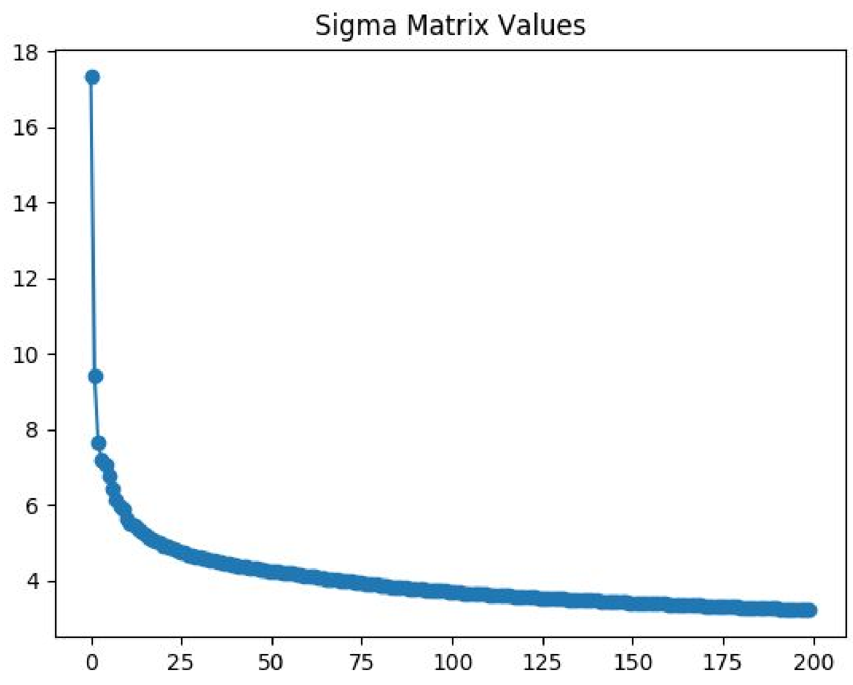
Analysis	Homogeneity	Completeness	ARI	AMI
PCA_log	0.347	0.397	0.117	0.344

NMF_log	0.320	0.387	0.087	0.317

Analysis	Homogeneity	Completeness	ARI	AMI
PCA_log w/norm	**0.393**	**0.410**	**0.236**	**0.391**
NMF_log w/norm	0.382	0.405	0.195	0.480

Form the chart above, we can see that **normalization after logarithmic PCA** still yields the best clustering score. However, the score is much smaller than that in the bipartite clustering. This actually makes sense because in this part we use larger TFIDF matrix and also more clusters.

Part F. Evaluate Topic-wise Clustering



In this part we first drew the sigma matrix values again. According to the graph, it seems that sigma value approaches to a stable value after k = 200 just as it was. We also found the best k value still locate in the range of 30 ~ 50 and the clustering results difference for k in this range is really slim. Therefore, we still choose 50 as our k value in this part. Also, the total number of classes is 6, we found there is no difference to adjust initial number of clustering to a higher value for better performance.

N_componenet = 50, init k for kmean clustering =10

Analysis	Homogeneity	Completeness	ARI	AMI
PCA	0.271	0.339	0.154	0.270
NMF	0.255	0.351	0.167	0.254

Analysis	Homogeneity	Completeness	ARI	AMI
PCA_norm	0.319	0.338	0.260	0.318
NMF_norm	0.371	0.403	0.274	0.371

Analysis	Homogeneity	Completeness	ARI	AMI
PCA_log	0.282	0.317	0.136	0.281
NMF_log	0.296	0.343	0.140	0.295

Analysis	Homogeneity	Completeness	ARI	AMI
PCA_log w/norm	0.338	0.376	0.319	0.338
NMF_log w/norm	**0.374**	**0.388**	**0.284**	**0.374**

According to tables above, we can see that **normalization after logarithmic NMF reduced TFIDF matrix** yields the best clustering score. Since we have larger database than before, we get lower score than Part C.