Project 2

Members:
Daphne Chiou
Nien-Jen Cheng
Hongyang Li

Files included in the folder:	
Q1.py	Q1
Q_2_v2.py	Q2
Q3.py	Q3, run PCA
Q3a.py	Q3, calculate variance ratios retained and plot figs
Q3b.py	Q3, calculate the 5 measures and contingencies of LSI and NMF
Q_4a.py	Q4(a), find the predict labels with best "r" and plot the scatter figs
Q_4b.py	Q4(b), calculate the logged, normed version of the PCA matrices
Q_4_plot.py	Q4(b), plot the scatter figs
Q_5.py	Q5
Project02_Report.pdf	This report

Q1

In this section, we do the same thing as in project one. Instead of generate the TF-TDF matrices for training and testing data, we merged the two together. In an unsupervised learning, we don't need to separate the two kinds.

The categories we included are:

```
'comp.graphics', 'comp.os.ms-windows.misc', 'comp.sys.ibm.pc.hardware',
'comp.sys.mac.hardware',
'rec.autos', 'rec.motorcycles', 'rec.sport.baseball', 'rec.sport.hockey'
```

We chose the minimum frequency to be 3, and we chose the stop word set as "text.ENGLISH_STOP_WORDS" in NLTK.

The dimension of the TF-TDF matrix was (7882, 27768)
(7882 documents and 27768 terms)

Q2

After constructing the TFIDF matrices, we now are capable to cluster the documents into two classes by applying k-means algorithm. Starting with random class labels assignment, the k-means algorithm calculates the centroids of each class and re-assign each document to its closest class based on Euclidean distance toward each centroid.

(a) We measure the results by contingency matrix, which is similar to the concept of confusion matrix instead Aij represents the amount of data points in class i that are assigned to cluster j.

	cluster 1	cluster 2
class 1	1	1559
class 2	623	967

We found that nearly all of the data points in class 1 are assigned to cluster 2, while only about ⅖ data points in class 2 are assigned to cluster 1. This implies the k-means algorithm trained on TFIDF tends to predict cluster 2 over cluster 1. Therefore, we consider the TFIDF matrices as the baseline result.

(b) To further measure the results, we use homogeneity score, completeness score, v measure score, adjusted rand score and adjusted mutual info score as suggested.

homogeneity_score	0.2267
completeness_score	0.3156
v_measure_score	0.2638
adjusted_rand_score	0.1483
adjusted_mutual_info_score	0.2265

Q3

In this section, we transform the term vector into the feature space via PCA. Two types of PCA will be implemented in this project. The LSI (SVD based) and the NMF.

(a)

In the previous project, we chose the length of the feature space to be 50. However, the length of the feature is a hyperparameter that can be used to optimize the result. In order to find the best length of the feature length, we execute the LSI with length, r from 1 to 1000. After this, we can find what is the percentage that the variance was retained of the truncated SVD.

<i>

We calculate the SVD with r =1000. By the function shown below.

```
def SVD(X_tfidf):
    from sklearn.decomposition import TruncatedSVD
    svd = TruncatedSVD(n_components=1000)
    W_train_svd = svd.fit_transform(X_tfidf)
    return W_train_svd
```

Then we take the first r columns of the PCA matrices and compare them with the original matrix. Then we calculate the retained variance.

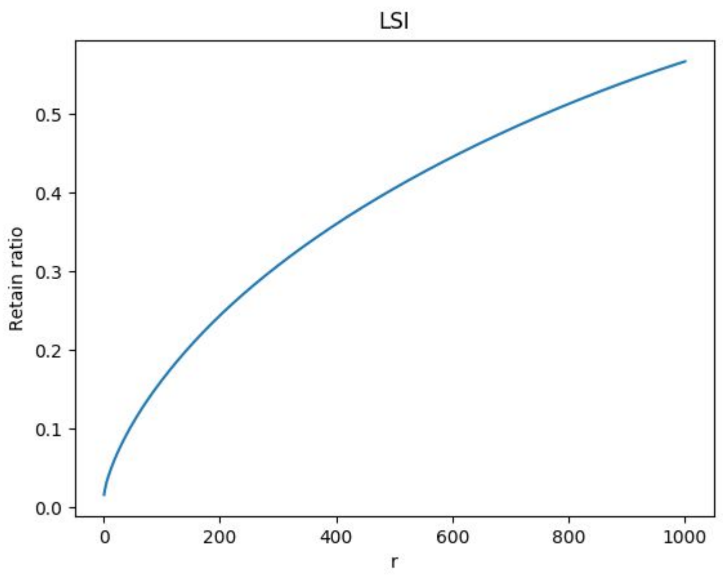
The variance of the matrices, X is the trace of X^TX or XX^T. The ratio of the retained variance is variance_retained / variance_totoal.

The function is defined as

```
def var_ratio(X_tfidf,W):
    var_tot = X_tfidf.toarray().T.dot(X_tfidf.toarray()).trace()
    ratio = np.zeros((1000,2))

    for r in np.arange(1000):
        X_retain = W[:,0:r+1]
        var_retain = np.trace( X_retain.T.dot(X_retain))
        var_ratio = var_retain / var_tot
        ratio[r] = [r+1,var_ratio]
    return ratio
```

The plot the ratio vs feature length, r is shown below. We can see that less than 60% of the variance is kept even with r =1000.



r	ratio
991	0.564578
992	0.564815
993	0.565052
994	0.565288
995	0.565524
996	0.56576
997	0.565995
998	0.56623
999	0.566465
1000	0.566698

<ii>
Now, we want to calculate the 5 measurements of each PCA matrices with the selected r=[1,2,3,5,10,20,50,100,300]. We will list them below. We can see that when r =2, we get the best result. It is inferred that the first 2 components of the features, which might be caused by the most frequent terms determine the categories. The rest of the data may be just noise, which interfere the classifier.

LSI

r	homogeneity	completeness	V_measure	adjusted_rand	adj_mut_info
1	0.00028431	0.000288884	0.000286578	0.00031746	0.000192785
2	0.608958	0.609259	0.609109	0.713498	0.608922
3	0.416446	0.450618	0.432859	0.419909	0.416392
5	0.221694	0.309964	0.258502	0.145156	0.221623
10	0.233915	0.320684	0.270512	0.156993	0.233845
20	0.235336	0.321754	0.271842	0.158607	0.235266
50	0.240369	0.324814	0.276283	0.165349	0.2403
100	0.243748	0.32809	0.279699	0.168252	0.243679
300	0.247946	0.330542	0.283348	0.174133	0.247877

NMF					
r	homogeneity	completeness	V_measure	adjusted_rand	adj_mut_info
1	0.00028431	0.000288884	0.000286578	0.00031746	0.000192785
2	0.592845	0.608067	0.600359	0.648592	0.592807
3	0.237561	0.3171	0.271628	0.169503	0.237492
5	0.187215	0.283748	0.225588	0.108544	0.187141
10	0.0959119	0.218751	0.133354	0.0359235	0.0958291
20	0.0919378	0.215337	0.128859	0.0333699	0.0918547
50	0.0698553	0.195404	0.102918	0.0206728	0.06977
100	0.0723913	0.195983	0.105729	0.022389	0.0723063
300	0.0146912	0.135602	0.135602	0.00137329	0.0146006

The contingency matrices are show below

r	LSI	NMF
1	2202 1701 2323 1656	2202, 1701 2323, 1656
2	3671 232 380 3599	731, 3172 3943, 36
3	3862 41 1346 2633	3890, 13 2305, 1674
5	3898 5 2434 1545	3898, 5 2637, 1342
10	3 3900 1603 2376	712, 3191 2, 3977
20	3 3900 1611 2368	685, 3218 2, 3977
50	3899 4 2334 1645	532, 3371 2, 3977
100	3900 3 2321 1658	3347, 556 3976, 3
300	3899 4 2292 1687	113 3790 0 3979

Q4

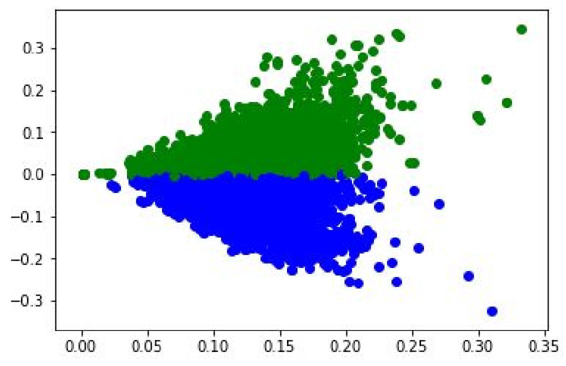
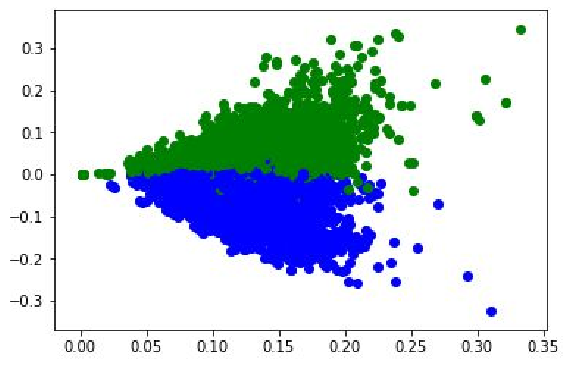
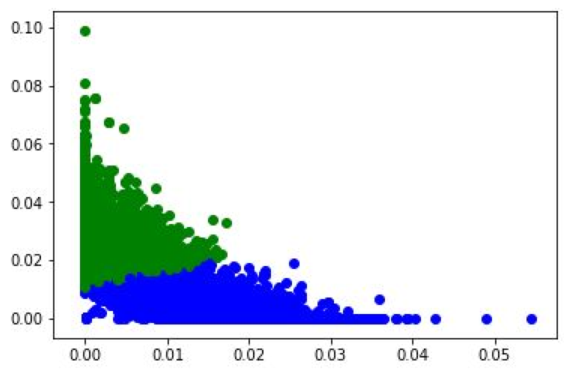
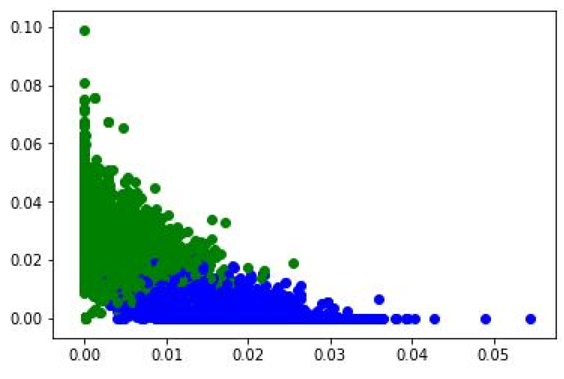
(a)

The color plots are shown below. The green and blue ones correspond the two classes. We can see that they were separated by a linear boundary. However, the margin is too close, we need to process some algorithms to make them more separated.

We can see the two group well-separated, there are some islands away from the continent. Although this happened, the algorithm can still mark them. Compared to the plot with true label. We see that there were some points cross the linear boundary. Therefore, although the PCA transformed quite well, it was not good enough.

Class_1 = ['comp.graphics', 'comp.os.ms-windows.misc', 'comp.sys.ibm.pc.hardware', 'comp.sys.mac.hardware']

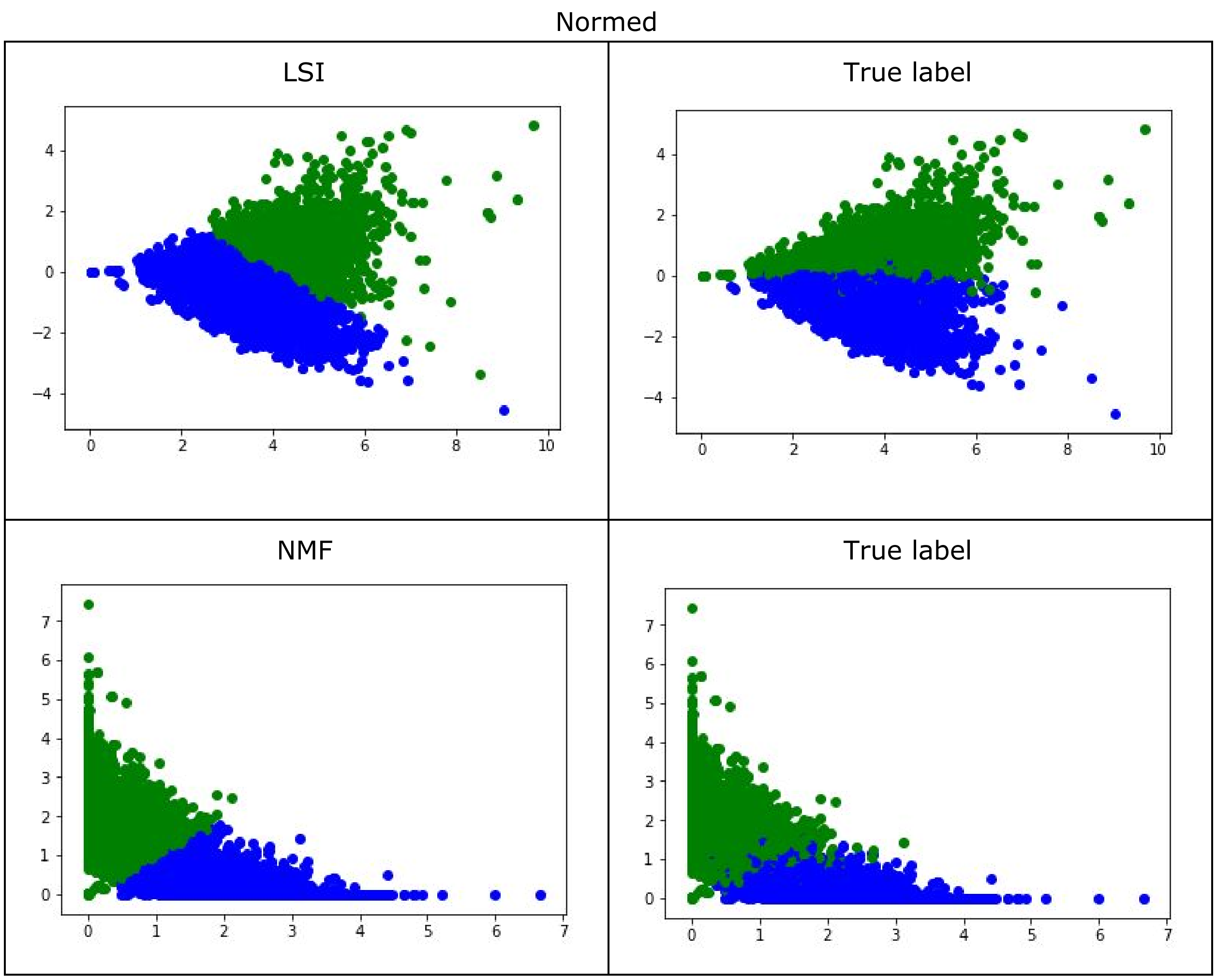
Class_2 = ['rec.autos', 'rec.motorcycles', 'rec.sport.baseball', 'rec.sport.hockey']

LSI	True label
	
NMF	**True label**
	

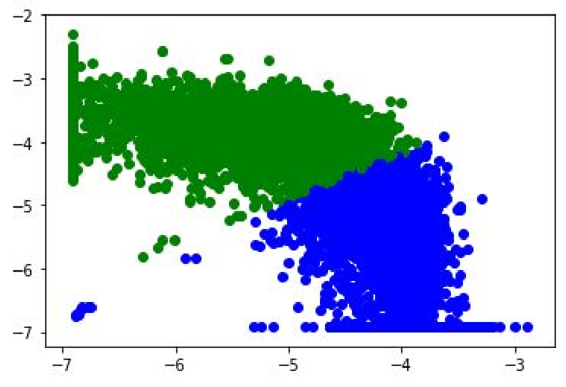
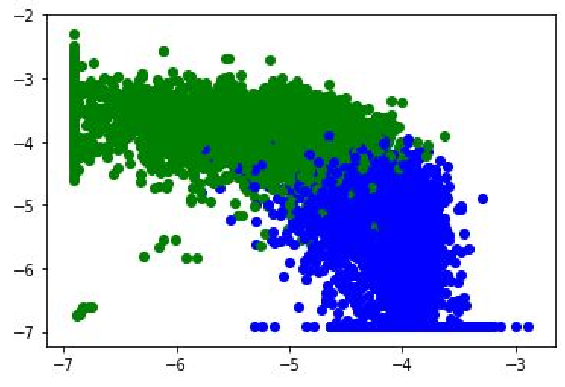
(b)

In this part, we apply normalization and non-linear transformation to the PCA matrices.

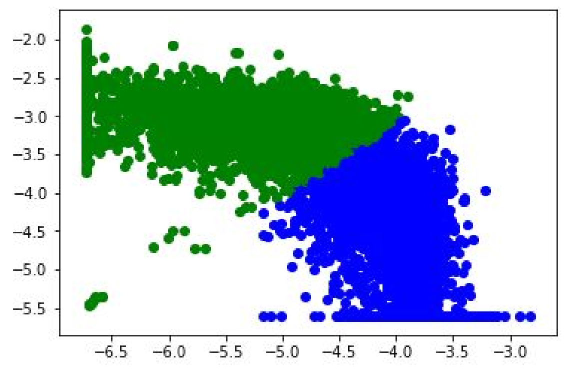
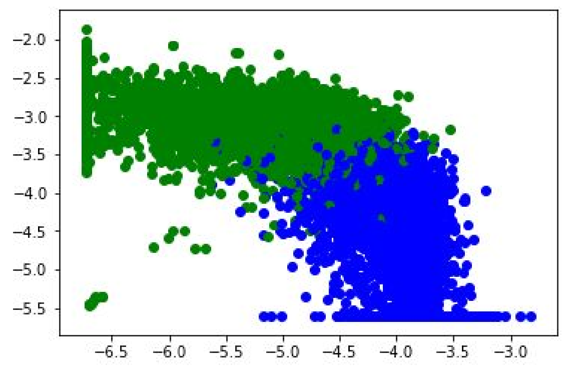
The normalization just normalizes each column in the matrices. We can see from the plots below that after normalization, there were points in LSI assigned to class 2. The boundary was tilted. However, in NMF, they have similar results.

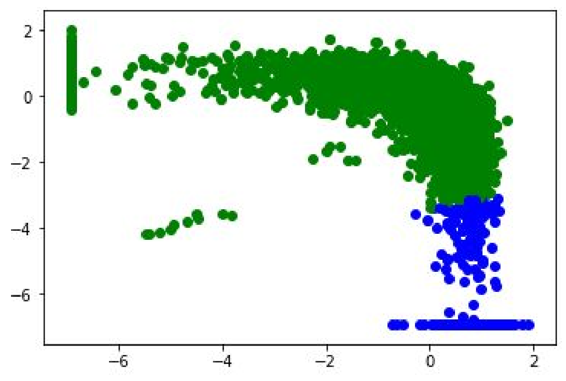
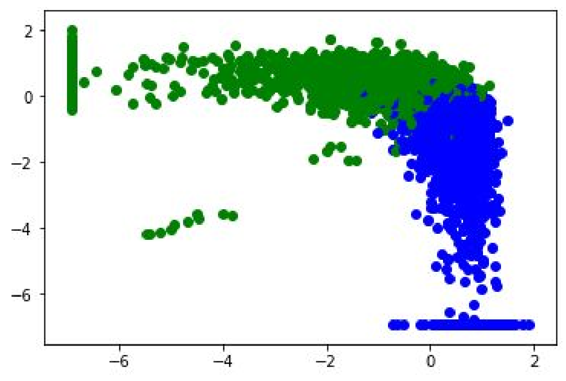


We use the logarithm as our non-linear transformation this time. The logarithm can apply on positive values, so we only implemented this on NMF. To prevent too small values in the log function, we add a bias 10-3 with the input matrices.From the graph below, we can see that the two clusters were more separated. The reason that the logarithm increase the clustering result is that the original values range from 0 to 1, which the range is small. Now, after the logarithm, the range was from 10-3 to 1, which was scratched. Therefore, we got a more separate result. Furthermore, if we norm the values before logarithm, the scatter will be slim.

Logarithm NMF	True label
	

NMF Logarithm and Norm

Logarithm + Norm	True label
	

Norm + Logarithm	True label
	

Q5

In this section, we further extend our experiment to a multiclass dataset. Instead of grouping the data into two major classes, we use the original label of each document, which falls into one of the 20 classes. Similar to what we have done with two-class clustering, we first remove the stopwords and transform the documents into the TFIDF matrix. Then, apply both SVD and NMF dimension reduction technique to represent the data in a more compact way. Due to the endless runtime, we experiment the r value only on [1, 2, 3, 5, 10, 20, 50, 100, 300]. We use different measurement method to evaluate the best r value, and found that different measurements agree on the r value. For example, the homogeneity are shown as below.

r	1	2	3	5	10	20	50	100	300
SVD	0.2238	0.2485	0.2931	0.3239	0.3771	0.3830	0.4041	0.4089	**0.4308**
NMF	0.0280	0.1679	0.2040	0.2764	**0.3188**	**0.3188**	**0.3188**	**0.3188**	**0.3188**

When applying SVD reduction, a larger r has significantly better results, which is pretty reasonable since the larger r is, the more information is perceived. The best r value is therefore the largest r=300, and the homogeneity is as high as 43%. On the other hand, the NMF reduction is not as sensitive to r as SVD is. The results of r = 10, 20, 50,100, 300 are all the same, implying that greater r does not guarantee a better result. The best homogeneity is around 31%. In the following section, we will stick with r=300 for SVD reduction and r=10 for NMF reduction.

Next, we apply different transformation methods on the data.

	homogeneity	completeness	v measure	rand score	mutul info
nmf_norm	0.3500	0.3589	0.3544	0.2052	0.3479
svd_norm	0.3594	0.4139	0.3847	0.1566	0.3573
nmf_log	0.2048	0.2068	0.2058	0.085	0.2023
nmf_log_norm	0.1854	0.1868	0.1861	0.0806	0.1828
nmf_norm_log	0.1846	0.1865	0.1855	0.0742	0.1819

Applying normalization on NMF reduction matrix generates a better result, 35% over 31% in terms of homogeneity. However, normalization deteriorates the results of SVD by ~8%.

Since SVD representation contains negative values, it is not suitable for logarithm transformation. We apply it only on NMF representation. The order of logarithm and

normalization is denoted by the order in the variable name. For instance, nmf_log_norm indicates that the logarithm is applied prior to the normalization.
Unfortunately, applying logarithm and normalization transformations on NMF matrix, regardless of the order, did not improve the results as shown in the table above. This is reasonable since both the logarithm and normalization only map the data points onto a new distribution and did not change the relations among data points.