Qiong Hu (405065032)
Daniel Ahn (504817439)
Riyya Hari Iyer (305427411)
Juan Piao (304959335)

Project 2 Report

Clustering

ECE 219
Data Mining
February 3, 2021

Introduction

Unsupervised Learning

Unsupervised Learning refers to one of three types of Learning in Artificial Intelligence (the other two being Supervised Learning and Reinforcement Learning). This refers to a type of learning where the given dataset does not include labels associated with any of the data points. Thus, the learning process in unsupervised learning does not have distinguished "train/test" process as in supervised learning. It would simply try to gauge the underlying pattern and features in the data points and make prediction of a label for each data point. To evaluate the performance of an unsupervised learning model, the loss would be the absolute difference between the predicted labels and the actual labels, for which there are five common measures: homogeneity score, completeness score, V-measure, adjusted rand score, and adjusted mutual info score.

Clustering algorithms belong to unsupervised methods, which differs from classification in that no *a priori* labeling or grouping information of the data points is available. One of common clustering algorithms is K-means clustering. And in this project, we will also investigate on other clustering algorithms, including Agglomerative clustering, DBSCAN, and HDBSCAN.

K-means Clustering

K-means clustering is one of the prominent models for Unsupervised Learning. To implement the method, K points are initially randomly assigned to the scattered dataset as cluster center points. During each iteration, any given data point in the dataset aligns itself to the nearest cluster center out of these K points. In other words, labels are assigned to data points based on the least distance among all the distances between the data point and all cluster center points. After updating the labels, K center points are re-placed to the new centre of their respective clusters, which is the mean of the data points that are currently within the cluster. The iteration would stop when the maximum iteration number is reached or the grouping result converges.

Part 1 - Clustering of Text Data

Basic Workflow

1. Building the TF-IDF matrix

QUESTION 1:

We removed the headers and footers when loading the full subset of eight categories in the "20 Newsgroup" dataset, and applied feature extraction functions "CountVectorizer" and "TfidfTransformer" to transform it into TF-IDF vectors. Parameter "min_df" was set to 3 and no stopword was applied. The resulting dimensions of the TF-IDF matrix is: 7882×23823.

2. K-means Clustering

QUESTION 2:

The next step is to create a K-Means clustering model that fits the TF-IDF matrix, and then we use the predicted labels for evaluation. The number of clusters chosen is 2 because the selected dataset is segregated into two categories: "Computer Technology" and "Recreational Activity".

The contingency matrix was plotted using function "contingency_matrix" from library "sklearn.metrics.cluster". Contingency matrix helps us see the comparison of difference between the predicted and actual labels as a table/matrix that includes information of false positives and false negatives.

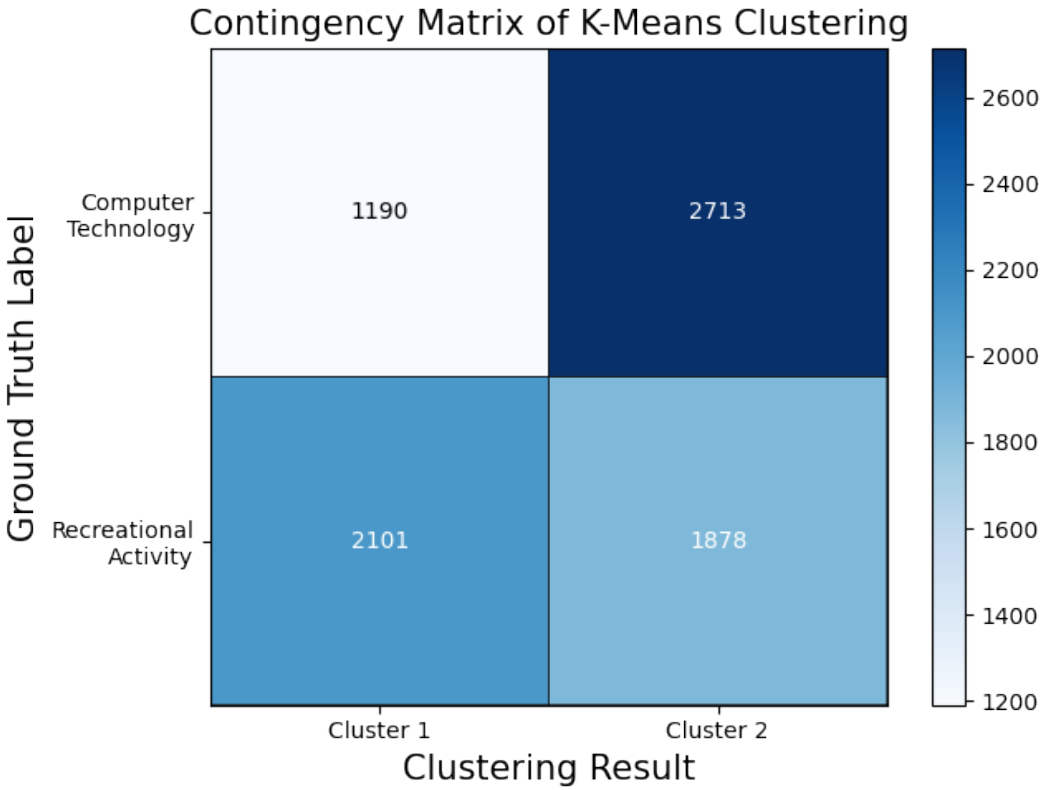

Figure 1: Contingency matrix of K-means clustering.

QUESTION 3:

We apply the following five measures to evaluate the prediction result of any given clustering algorithm[1]:

1. **Homogeneity** measures the overall consistency of ground truth labels within any single cluster.

2. **Completeness** measures the consistency of cluster labels within each of the ground truth classes.

3. **V-measure** is defined as the harmonic average of homogeneity and completeness score.

4. **Adjusted Rand Index** describes the similarity between the predicted labels and ground truth labels, which is similar to accuracy measure.

5. **Adjusted Mutual Information Score** measures the mutual information between the two distributions of the predicted labels and ground truth labels.

All five measure scores are within the range of 0 to 1, with 1 being the best performance and 0 being the worst. As we can see, homogeneity and completeness describe how well the clustering results align with the ground truth labels from two aspects, which can be a trade-off in certain circumstances. A good clustering model should achieve relatively high score for all five measures.

For implementation, all the measurement functions are imported from the library "sklearn.metrics". We apply them on the K-means clustering result we get from the model in Question 2, and the evaluation result is reported in Table 1.

Table 1: Evaluation results of K-means clustering.

Homogeneity	Completeness	V-measure	Adjusted Rand Index	Adjusted Mutual Info Score
0.0373	0.0380	0.0377	0.0490	0.0376

The low scores for all five measures in Table 1 show that the clustering performance is terrible. And the reason is that the test model is applied directly on the sparse TF-IDF matrix that has more than twenty thousand features. In order to improve the clustering performance, it is necessary to apply dimensionality reduction methods to extract a reduced subset of the matrix that includes mostly useful features.

In the following sections, we investigate on different dimensionality reduction methods, including NMF, SVD, and UMAP.

Dimensionality Reduction and Data Transformation

1. Dimensionality Reduction (SVD & NMF)

QUESTION 4:

We use the property "explained_variance_ratio_" of "TruncatedSVD" model to find the retained variance percentage under each r, and plot the accumulated variance percentage as in Fig. 2.

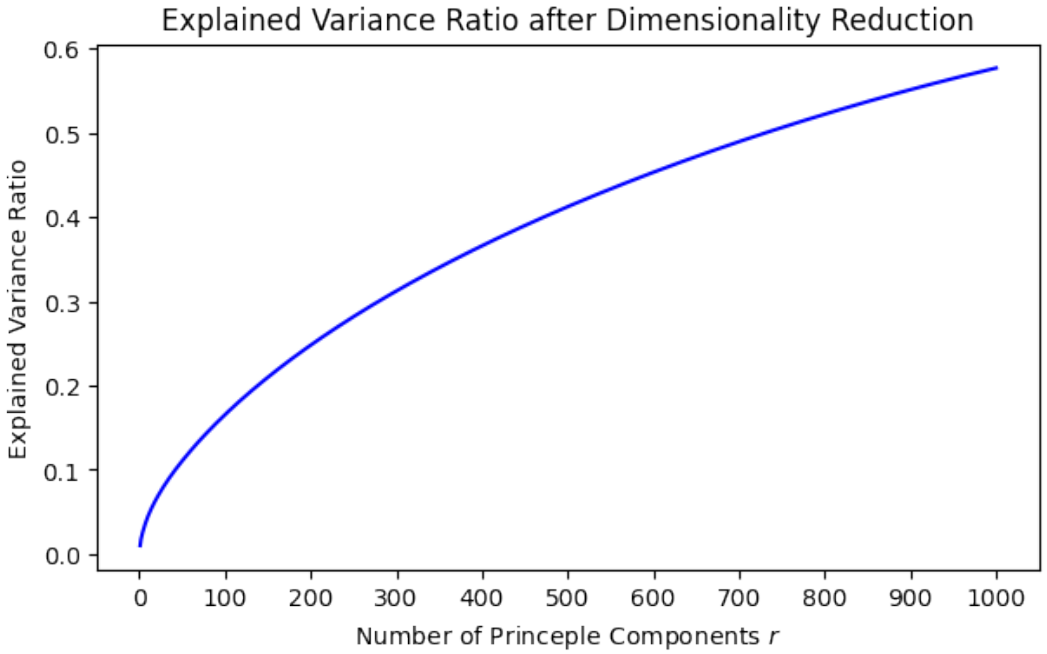

Figure 2: The percentage of variance that the top r principal components can retain.

From Fig. 2, we notice that the explained variance ratio increases as the number of principal components included increases. The proportion of variance explained by including only the first 100 principal component is about 15% and the proportion of variance explained by including the first 1000 principal component is about 58%. The trend of the curve looks similar to logarithmic curve.

QUESTION 5:

We run K-Means clustering algorithm multiple times on different reduced matrices using different r in Truncated SVD and NMF, and plot the measure scores in Fig. 3 and Fig. 4, respectively.

From the figures, we can find that:

Best n_components for SVD: $r = 2$

Best n_components for NMF: $r = 3$

Table 2: Evaluation results of K-means clustering using SVD and NMF with respective best r.

Algorithm	Homogeneity	Completeness	V-measure	Adjusted Rand Index	Adjusted Mutual Info Score
SVD	0.0611	0.0638	0.0625	0.0768	0.0624
NMF	0.4796	0.4840	0.4818	0.5763	0.4817

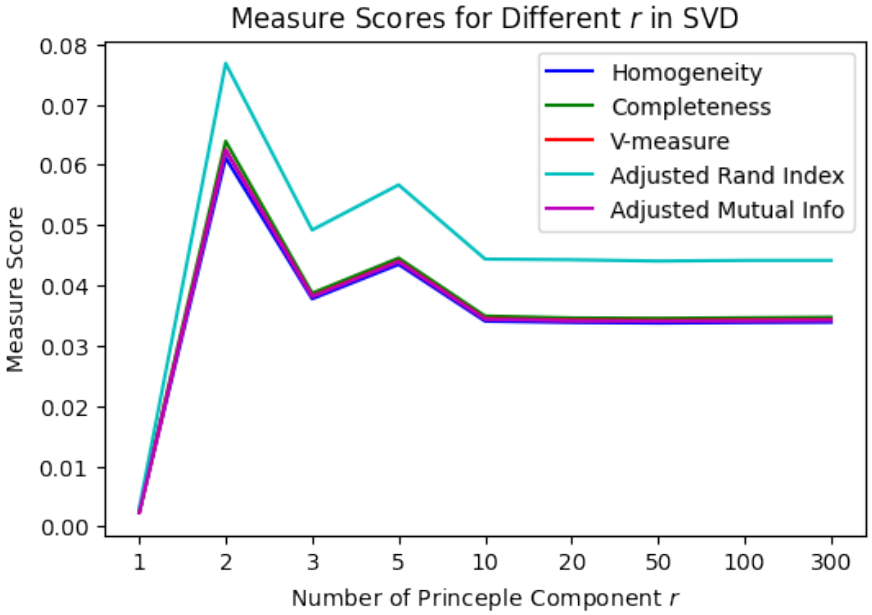

Figure 3: Measure scores for different r in SVD.

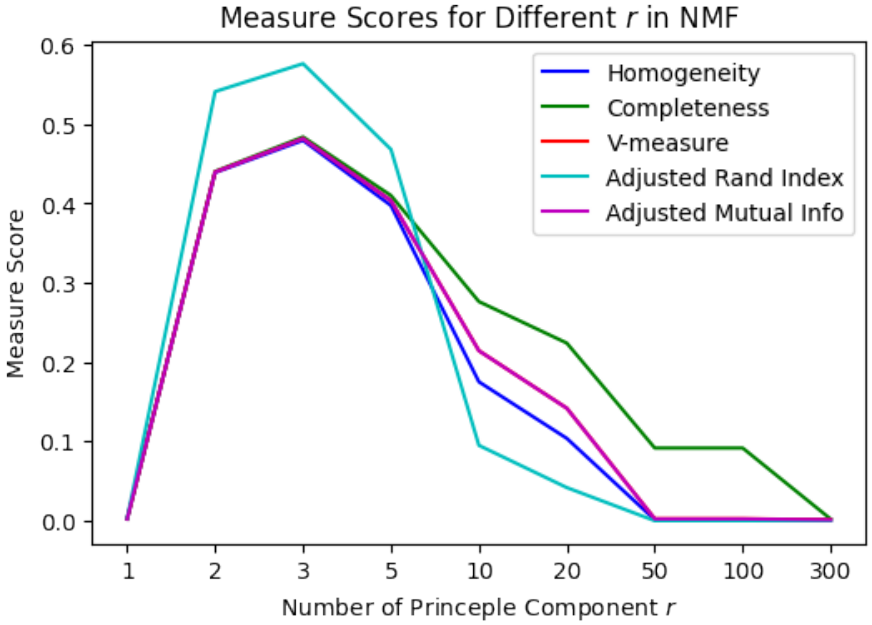

Figure 4: Measure scores for different r in NMF.

We notice that the five measures tend to have similar trend, among which V-measure and Adjusted Mutual Info often returns data that is so close that the two curves are almost indistinguishable in Fig. 3 and Fig. 4.

QUESTION 6:

From Fig. 3 and Fig. 4 we notice that the performance of the model first increases as r increases up until $r = 2$ (in SVD) or $r = 3$ (in NMF), and then has a general tendency of decreasing. The reason of the non-monotonic behavior of the measures is that, increasing the number of principal components retains more information about the original data (as shown in Fig. 2), hence the model performance increases as r increases at first.

But the K-means model performance starts to suffer soon due to the increased dimensionality, mainly because K-means algorithm clusters data points by optimizing the squared Euclidean distance of the points with the center of the corresponding clusters. But difference of euclidean distance between clusters would gradually become indistinguishable when the data points are scattered in high dimensions.

Therefore, the overall performance of the measures as r increases is a trade-off between preserved information and better performance of K-Means in lower dimensions, leading to a non-monotonic behavior.

2. Visualization

QUESTION 7:

We can visualize the clustering results by projecting the dim-reduced data points onto 2-D plane with 2-cluster dimensionality reduction methods, and coloring the points according to their labels.

Instead of using Truncated SVD for 2-D projection as instructed, we used PCA for projection, which follows the same projection methodology and also returns beautiful visualization plots as shown in Fig. 5 and Fig. 6.

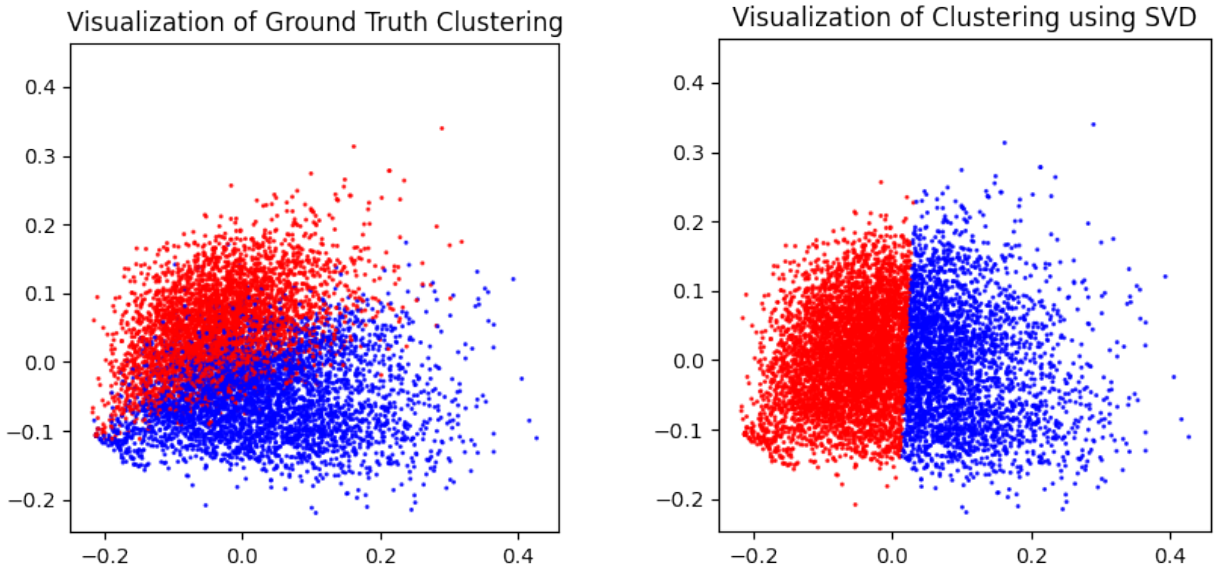

Figure 5: Visualization of the clustering result using SVD ($r = 2$).

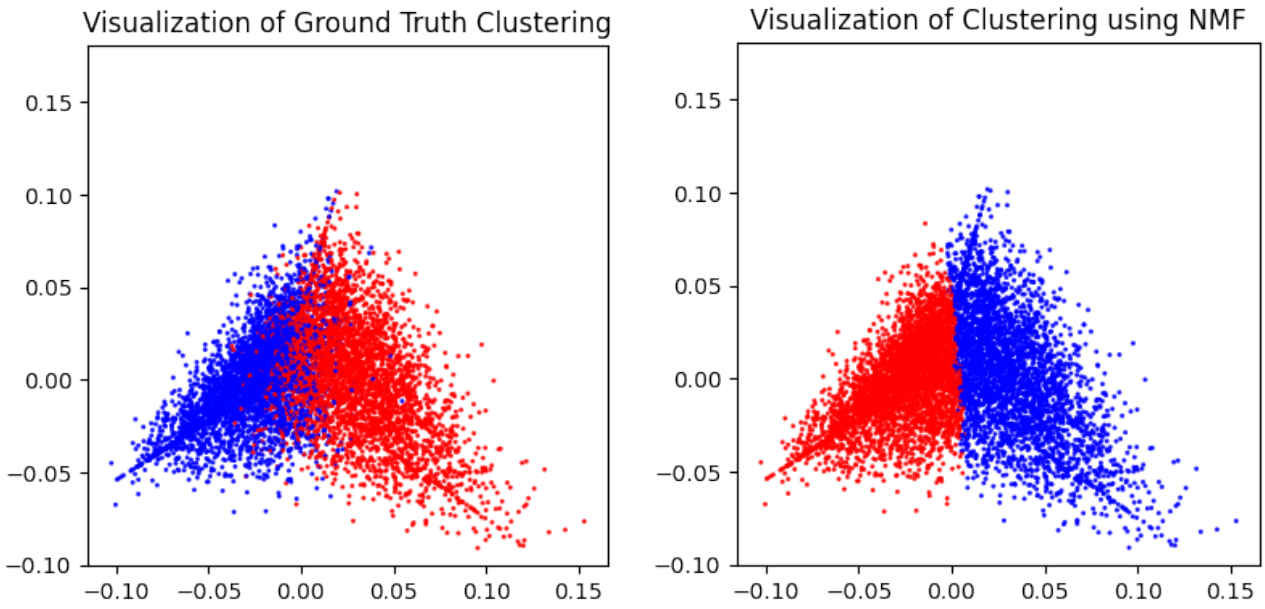

Figure 6: Visualization of the clustering result using NMF ($r = 3$).

QUESTION 8:

For dataset reduced by SVD with $r = 2$ (see Fig. 5), we notice that in the visualization of the ground truth clustering, the data distribution of each cluster is not spherical, and there are many outliers in both clusters. There are also many overlapping area in the edge of two clusters. The clustering result using K-Means algorithm has a quite clear boundary between the two clusters, and the shape of the boundary is skewed compared to the ground truth. The visualization confirms that K-Means clustering does not work well with SVD-reduced dataset.

The reason is that, K-Means algorithm assumes the variance of the distribution of each attribute is spherical and it is sensitive to outliers, therefore, the SVD-reduced data is not well suitable for K-Means algorithm.

For dataset reduced by NMF with $r = 3$ (see Fig. 6), from the clustering result we observe that the data is reasonably well separated by two clusters but with some points on the boundaries misclassified. This is because there are some overlapping area in the edge of two clusters as shown in the visulization of the ground truth data. The clusters are in triangular shapes, but they are roughly the same size in feature space and can be approximately wrapped in circles of the same size. Overall, the NMF-reduced data distribution is more suitable for K-Means clustering.

3. Clustering all 20 categories

QUESTION 9:

We load all 20 categories from the "20 Newsgroup" dataset and use same vectorizer method as in Question 1 to get TF-IDF matrix. The dimensions of the sparse matrix is: 18846×45674.

Now, the new target of the clustering task is to cluster the full dataset into 20 clusters that correspond to the given labels of 20 categories.

Using the same method as in Question 5, we run K-means clustering algorithm multiple times with different r for NMF and SVD (naturally choosing 20 as the number of components), and try to find a proper r to do the dimensionality reduction for retaining just the important components. An example figure of such experiment is shown in Fig. 7.

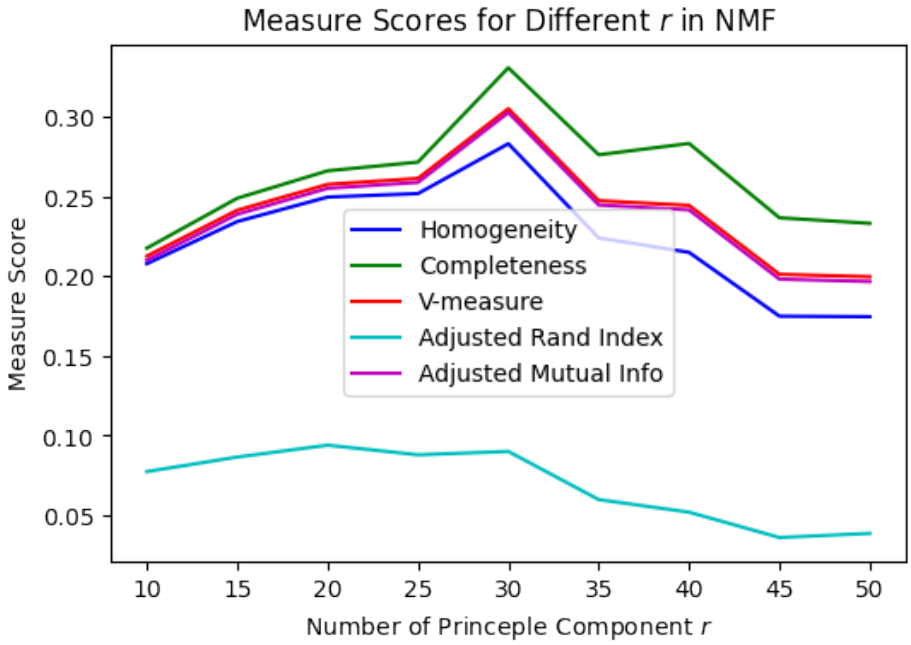

Figure 7: Measure scores for different r in NMF for all 20 categories.

We find that the best performance is achieved when reducing the dimension using NMF with $r = 30$. The potential reason why NMF-reduced dataset performs better than SVD is already revealed in Question 8, which is that the NMF-reduced data distribution is more ideal for K-Means algorithm.

We then use the properly reduced matrix to perform K-Means clustering with "n_components=20". The contingency matrix of the result is shown in Fig. 8, and the five measures are reported in Table 3.

Table 3: Evaluation results of K-means clustering on all 20 categories (using NMF, $r = 30$).

Homogeneity	Completeness	V-measure	Adjusted Rand Index	Adjusted Mutual Info Score
0.2829	0.3304	0.3048	0.0901	0.3024

Note that for convenience of observation, we already did the permutation for the contingency matrix in Fig. 8 by using function "linear_sum_assignment".

Qiong Hu (405065032)
Daniel Ahn (504817439)
Riyya Hari Iyer (305427411)
Juan Piao (304959335)

Project 2 Report

Clustering

ECE 219
Data Mining
February 3, 2021

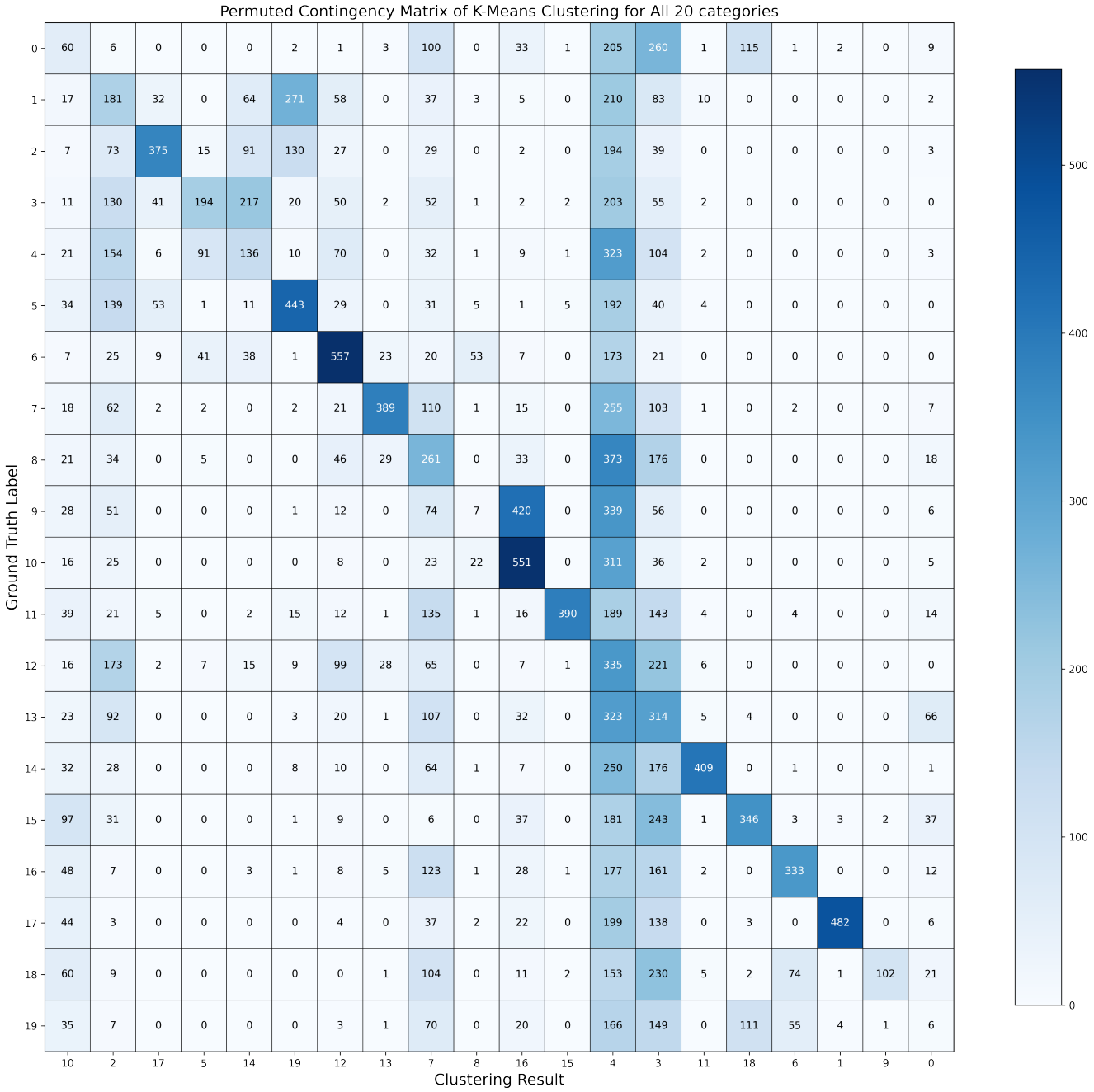

Figure 8: Permuted contingency matrix on all 20 categories (using NMF, $r = 30$).

From Fig. 8, we notice that most of the large numbers are on the diagonal line of the permuted contingency matrix, showing that the clustering algorithm roughly returns an acceptable clustering result. However, there are also some clusters that have plenty misclassified data, such as clusters with predicted index number of 3 and 4.

QUESTION 10:

For this question, we try Kullback-Leibler Divergence for NMF, and compare the performance with Frobenius norm. KL Divergence is a way of viewing probability distribution.

We use the same method as in Question 5 to look for best r, and the result is as follows:

Best n_components for NMF with Frobenius norm: $r = 70$

Best n_components for NMF with KL Divergence: $r = 20$

The five evaluation metrics of both methods with respective best r is reported in Table 4 for comparison.

Table 4: Evaluation results of K-means clustering on all 20 categories with different cost function.

Cost Function	Homogeneity	Completeness	V-measure	Adjusted Rand Index	Adjusted Mutual Info Score
Frobenius	0.2607	0.2940	0.2763	0.0901	0.2739
KL Divergence	0.4296	0.4396	0.4345	0.2360	0.4327

From Table 4, we notice that KL Divergence significantly improves all measures.

Note that both Frobenius norm and KL Divergence belong to the cost function family of β-divergence, where Frobenius norm is $\beta = 2$ and KL Divergence is $\beta = 1$. The reason why Frobenius norm is not the theoretically most reasonable choice for this dataset is that, it corresponds to the maximum likelihood estimator in the presence of additive i.i.d. Gaussian noise. However, the "20 Newsgroup" dataset is very sparse in which case Gaussian noise is not appropriate. Instead, the Poisson distribution is particularly well suited for integer-valued dataset, such as in our case, the documents are represented by vector of word counts[6]. Therefore, using NMF with KL Divergence as cost function would greatly improve the clustering performance.

4. Dimensionality Reduction (UMAP)

QUESTION 11:

We import function "UMAP" from the library "umap" to implement the UMAP tool for dimensionality reduction, and use the same method as in Question 5 to find the best r for two distance metrics in UMAP respectively:

Best n_components for UMAP with euclidean distance: $r = 2$

Best n_components for UMAP with cosine distance: $r = 100$

Then we use their respective best r to generate reduced matrix for K-Means clustering, and evaluate the clustering results. The five evaluation metrics are shown in Table 5, and the two permuted contingency matrices are shown in Fig. 9 and Fig. 10.

Qiong Hu (405065032)
Daniel Ahn (504817439)
Riyya Hari Iyer (305427411)
Juan Piao (304959335)

Project 2 Report

Clustering

ECE 219
Data Mining
February 3, 2021

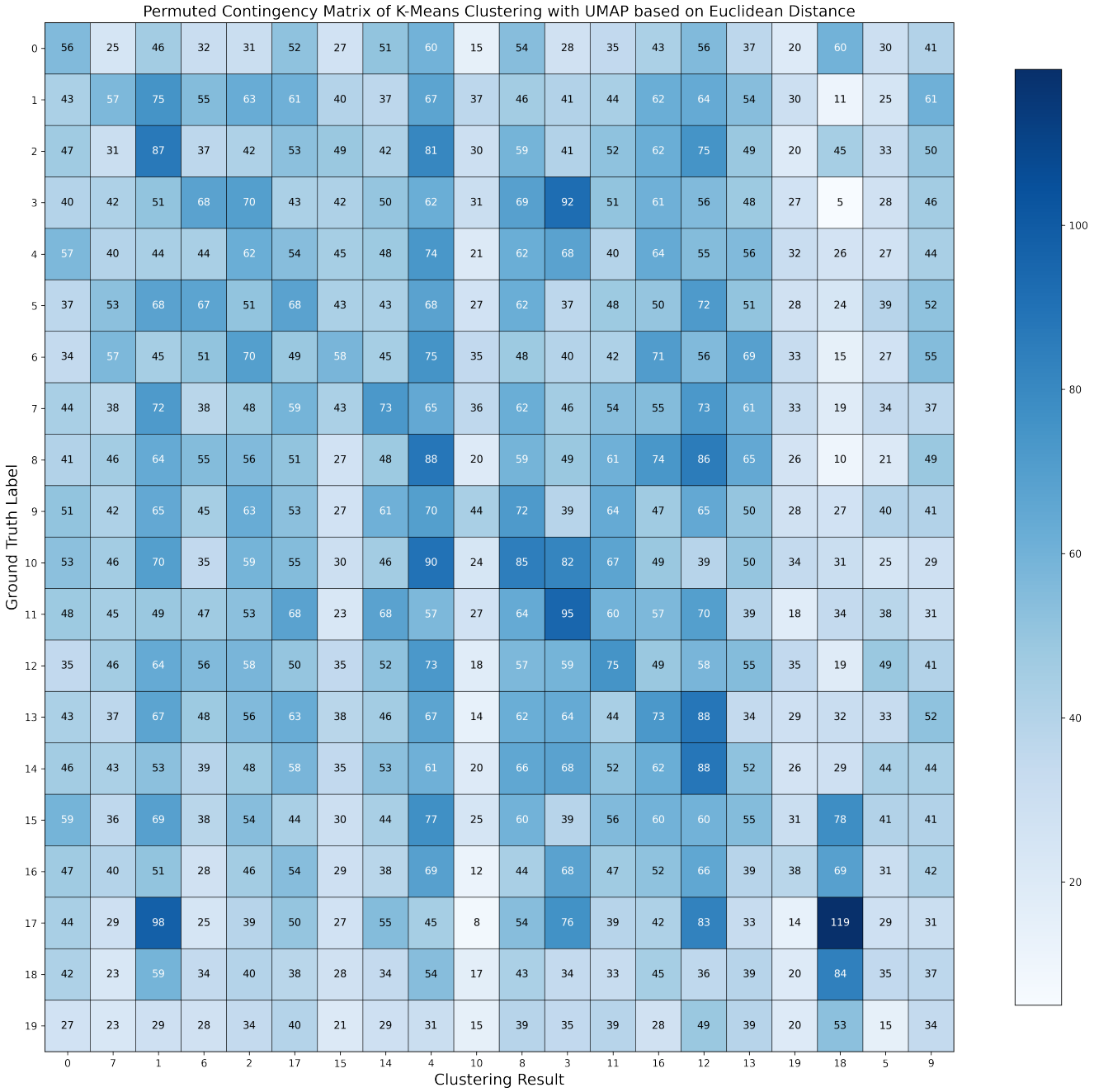

Figure 9: Permuted contingency matrix of K-Means clustering using UMAP with euclidean distance.

Qiong Hu (405065032)
Daniel Ahn (504817439)
Riyya Hari Iyer (305427411)
Juan Piao (304959335)

Project 2 Report

Clustering

ECE 219
Data Mining
February 3, 2021

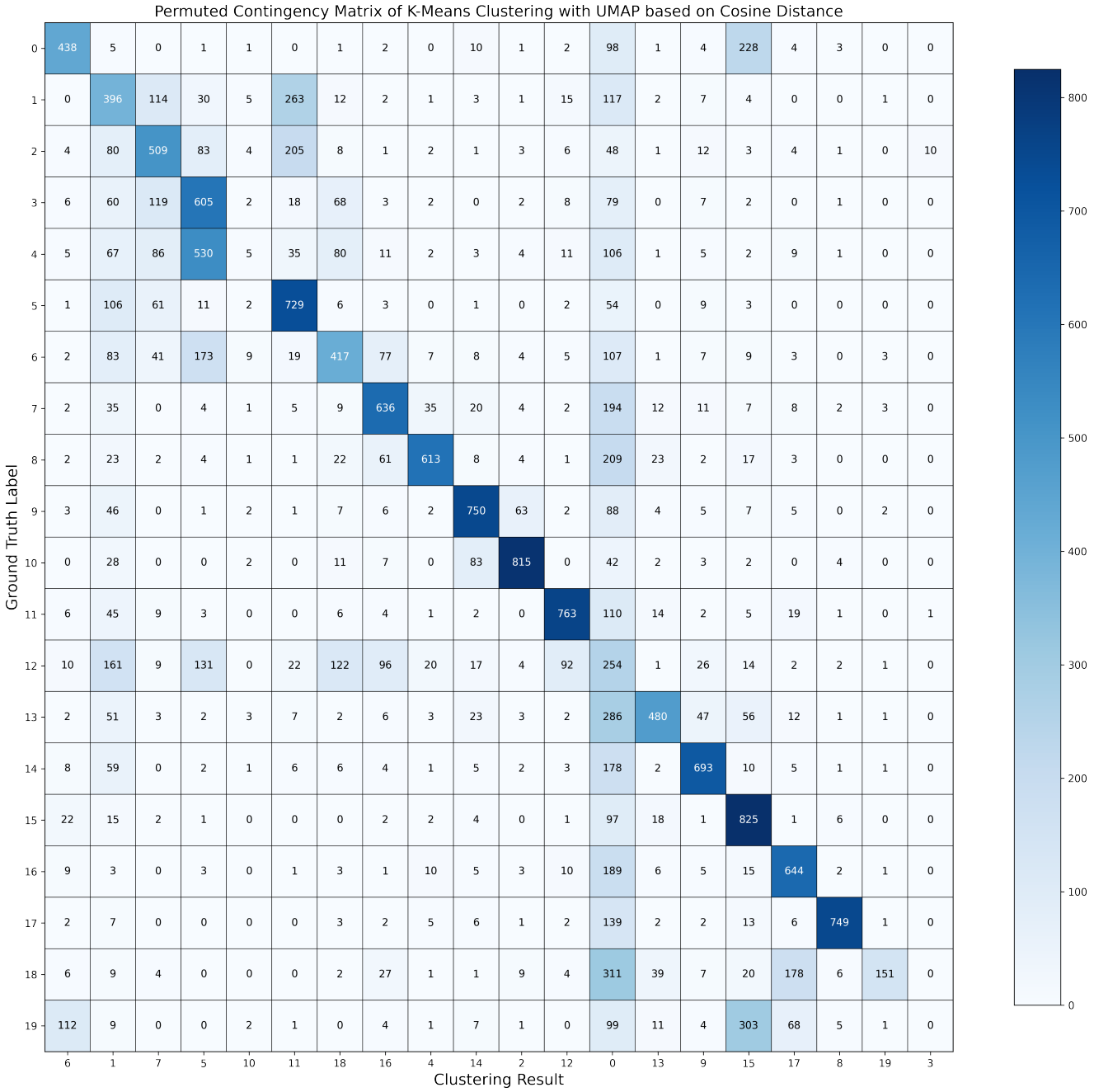

Figure 10: Permuted contingency matrix of K-Means clustering using UMAP with cosine distance.

Table 5: Evaluation results of UMAP using "euclidean" and "cosine" distance.

Metric	Homogeneity	Completeness	V-measure	Adjusted Rand Index	Adjusted Mutual Info Score
Euclidean	0.0105	0.0106	0.0105	0.0022	0.0073
Cosine	0.5025	0.5397	0.5204	0.3346	0.5189

From Table 5, we observe that "Cosine" metric produces much better results than "Euclidean" metric for UMAP. This is due to the fact that by using "Euclidean" metric we calculate the distance which takes the length of documents into account, and by using "Cosine" metric, we focus on the orientation difference of the TF-IDF vectors. So two documents with a similar topic and different lengths may be apart by a huge euclidean distance but close in cosine distance. Therefore, using "Cosine" metric would have better clustering results for document clustering tasks[4].

QUESTION 12:

From two contingency matrices in Fig. 9 and Fig. 10, we notice that when using "Euclidean" distance, the contingency matrix looks like a mess, and almost all the clusters are confused with each other, while for "Cosine" distance, the highest values are almost all in the diagonal line of the contingency matrix. In order to explain these observations, we plot the visualizations of two clustering results in Fig. 11 and Fig. 12 for better understanding.

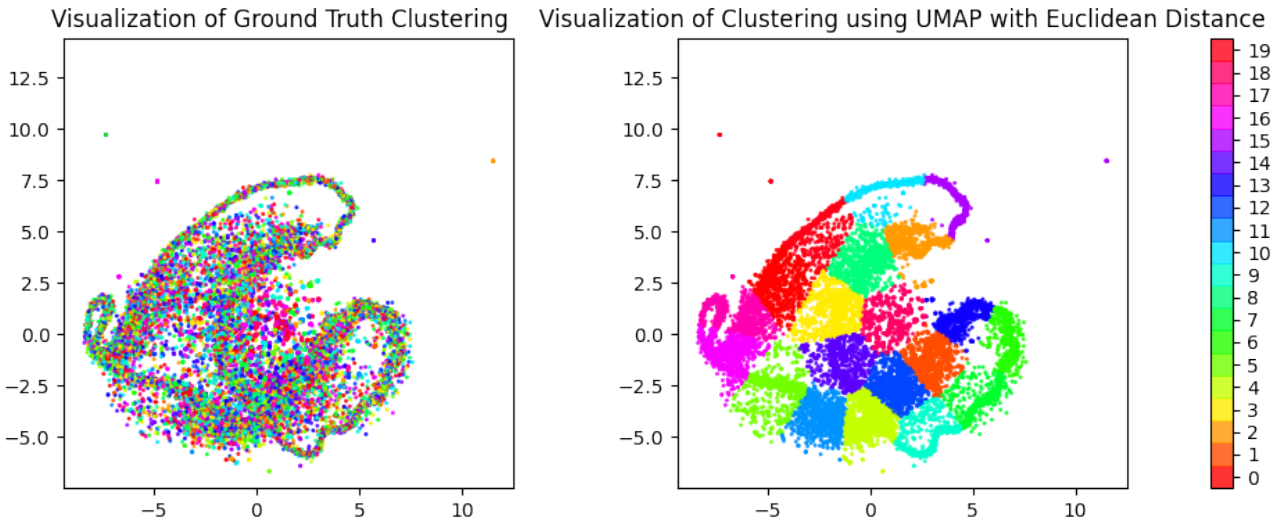

Figure 11: Visualization of the clustering result using UMAP with euclidean distance.

It is clearly shown in Fig. 11 that, when using "Euclidean" distance, the predicted clusters tend to be in the same neighborhood, which is very different from the ground truth, and hence it explains the poor performance in the five measures and mixed clusters in the contingency matrix.

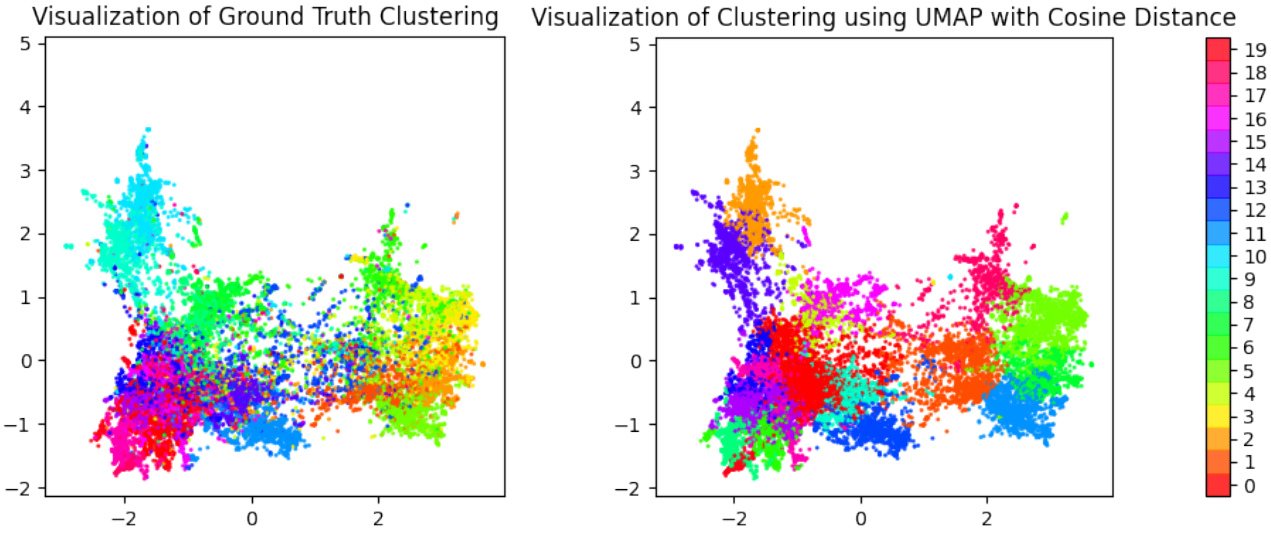

Figure 12: Visualization of the clustering result using UMAP with cosine distance.

As for UMAP using "Cosine" distance, the categories that are prone to be confused with each other are as follows (the index number corresponding to the ground truth labeling is included in the brackets after each label name):

- alt.atheism (0), soc.religion.christian (15), talk.religion.misc (19)
- comp.graphics (1), comp.os.ms-windows.misc (2), comp.windows.x (5), sci.electronics (12)
- comp.sys.ibm.pc.hardware (3), comp.sys.mac.hardware (4), misc.forsale (6), sci.electronics (12)
- rec.autos (7), sci.electronics (12)
- sci.electronics (12), talk.politics.guns (16), talk.politics.misc (18)

The result makes sense. For example, it is not uncommon to talk about "atheism" and "christian" in the same document that is related to "religion"; articles that talk about "electronics" that are "forsale" are very likely to mention "ibm" and "mac"; etc.

More Clustering Algorithms

1. Agglomerative Clustering

QUESTION 13: We use the model "AgglomerativeClustering" from the library "sklearn.cluster" to implement Agglomerative Clustering algorithm. The five clustering evaluation metrics for the performance of "ward" and "single" linkage criteria are reported in Table 6.

Table 6: Evaluation results of Agglomerative clustering using "ward" and "single" linkage criteria.

Linkage Criteria	Homogeneity	Completeness	V-measure	Adjusted Rand Index	Adjusted Mutual Info Score
Ward	0.4778	0.5187	0.4974	0.3118	0.4957
Single	0.0135	0.2689	0.0256	0.00004	0.0196

The linkage criterion determines which distance to use between sets of observation, and the algorithm will merge the pairs of cluster that minimize this criterion: "ward" uses the minimum of the variance of the clusters being merged, while "single" uses the minimum of the distances between all observations of the two sets[5].

From Table 6 we can see that the performance of "ward" is a lot better than that of "single". And by observing the visualization of the clustering, we notice that for "single", most of the data point is clustered into one big cluster and the rest 19 clusters are comparatively quite small with only a few points, which makes sense because in this way the single distance would be minimized, but it's not very effective for clustering our dataset.

2. DBSCAN and HDBSCAN

QUESTION 14:

In implementation, we import model "DBSCAN" from library "sklearn.cluster" and model "HDBSCAN" from an external library called "hdbscan", and experiment on hyperparameter "*eps*" and "*min_samples*".

The experiment is designed in this way:

- To investigate the effects of *eps*, we plot five measure scores with different *eps* sweeping from 0.02 to 0.5 with a step of 0.2, when *min_samples* is fixed to a small value (set to 5) or a large value (set to 100).
- To investigate the effects of *min_samples*, we plot five measure score with different *min_samples* from the list $[5, 10, 25, 40, 50, 60, 75, 100, 125, 150, 200]$, when *eps* is fixed to a small value (set to 0.2) or a large value (set to 0.5). These candidate *min_samples* values are empirically decided.

The complete experiment results include eight figures (four for DBSCAN and four for HDBSCAN) and quantitative outputs, which can be found in the code session in the attached printout of the Jupyter Notebook. Here we show four figures out of eight for demonstration.

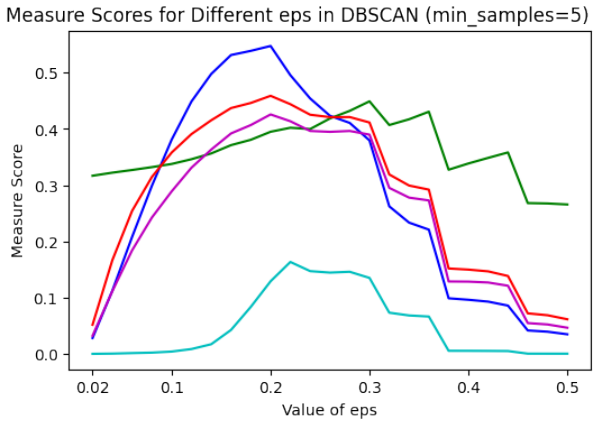

Figure 13: Effects of *eps* in DBSCAN

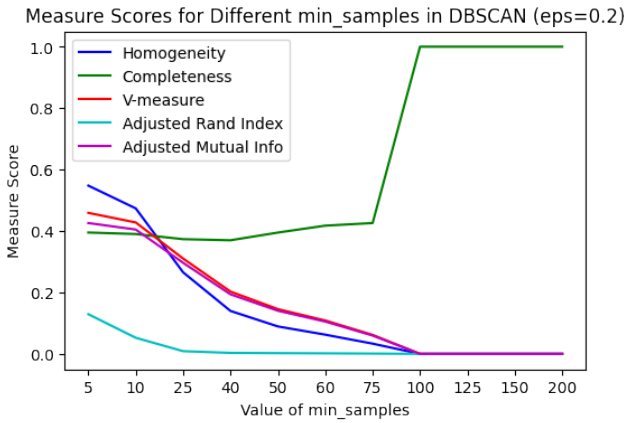

Figure 14: Effects of *min_samples* in DBSCAN

From Fig. 13, we find that the overall clustering performance first increases and then decrease when value of *eps* increases in DBSCAN while *min_samples* is set to 5. The best performance is reached when $eps = 0.2$.

In Fig. 14, the DBSCAN clustering performance continuously decreases when *min_samples* increases and *eps* set to 0.2. When $min\_samples > 100$, all the data points are clustered into one big cluster, leading to $completeness = 1$ and all the rest measures being 0, which indicates that the clustering algorithm becomes ineffective.

Therefore, by considering the combined effects of *eps* and *min_samples* together, the best DBSCAN hyperparameter combination we found is: $eps = 0.2$, $min\_samples = 5$.

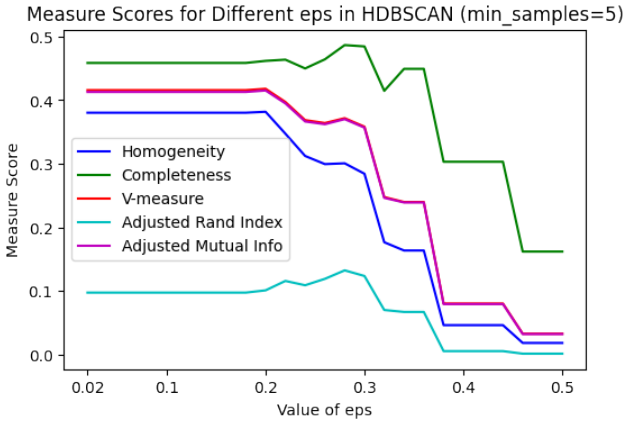

Figure 15: Effects of *eps* in HDBSCAN

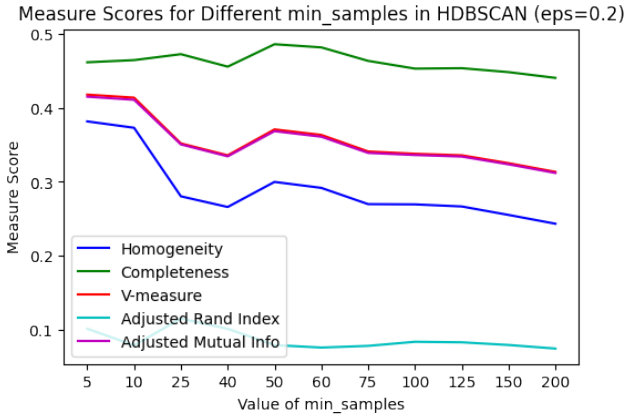

Figure 16: Effects of *min_samples* in HDBSCAN

As for HDBSCAN, we set *min_cluster_size* to 100 in all the experiments as instructed. When *min_samples* is set to 5 and *eps* sweeps from 0.02 to 0.5, the Completeness score, V-measure score, and Adjusted Mutual Information score remains same until $eps = 0.2$ and then drops; the

Completeness score and Adjusted Rand Index score slightly increases until *eps* = 0.28 and then also drops rapidly (as shown in Fig. 15)

The increment of *min_samples* when *eps* is set to 0.2 causes all five measures of HDBSCAN clustering result have a stable tendency of decreasing as shown in Fig. 16. Therefore, the best performance is when *min_samples* equals to 5.

In conclusion, the best hyperparameter combination that we found for both DBSCAN and HDBSCAN is the same: eps = 0.2, min_samples = 5. The five measures using these hyperparameters are reported in Table 7.

Table 7: Evaluation results of DBSCAN and HDBSCAN.

Algorithm	Homogeneity	Completeness	V-measure	Adjusted Rand Index	Adjusted Mutual Info Score
DBSCAN	0.5477	0.3951	0.4590	0.1291	0.4259
HDBSCAN	0.3818	0.4619	0.4181	0.1009	0.4153

QUESTION 15:

From Table 7, we can see that the best clustering result from DBSCAN has better performance than HDBSCAN in four out of five measures. The reason why DBSCAN has a lower completeness score is that, the DBSCAN model gives 416 clusters while the HDBSCAN model gives 26 clusters (both larger than the original 20 categories), thus the resulting clusters from DBSCAN has a lower possibility to completely include all the data points that originally belong to a same category.

The clustering label "-1" means that these data points are classified as noise by the model[7].

We choose DBSCAN with eps=0.2 and min_samples=5 as our best model, and plot the contingency matrix in Fig. 17.

By observing the contingency matrix, we notice that some categories are prone to be confused with each other, which are:

- alt.atheism (0), soc.religion.christian (15), talk.religion.misc (19)
- comp.sys.ibm.pc.hardware (3), comp.sys.mac.hardware (4)
- rec.sport.baseball (9), rec.sport.hockey (10)

Similar to the analysis in Question 12, and by using common sense, we can see that the result makes sense.

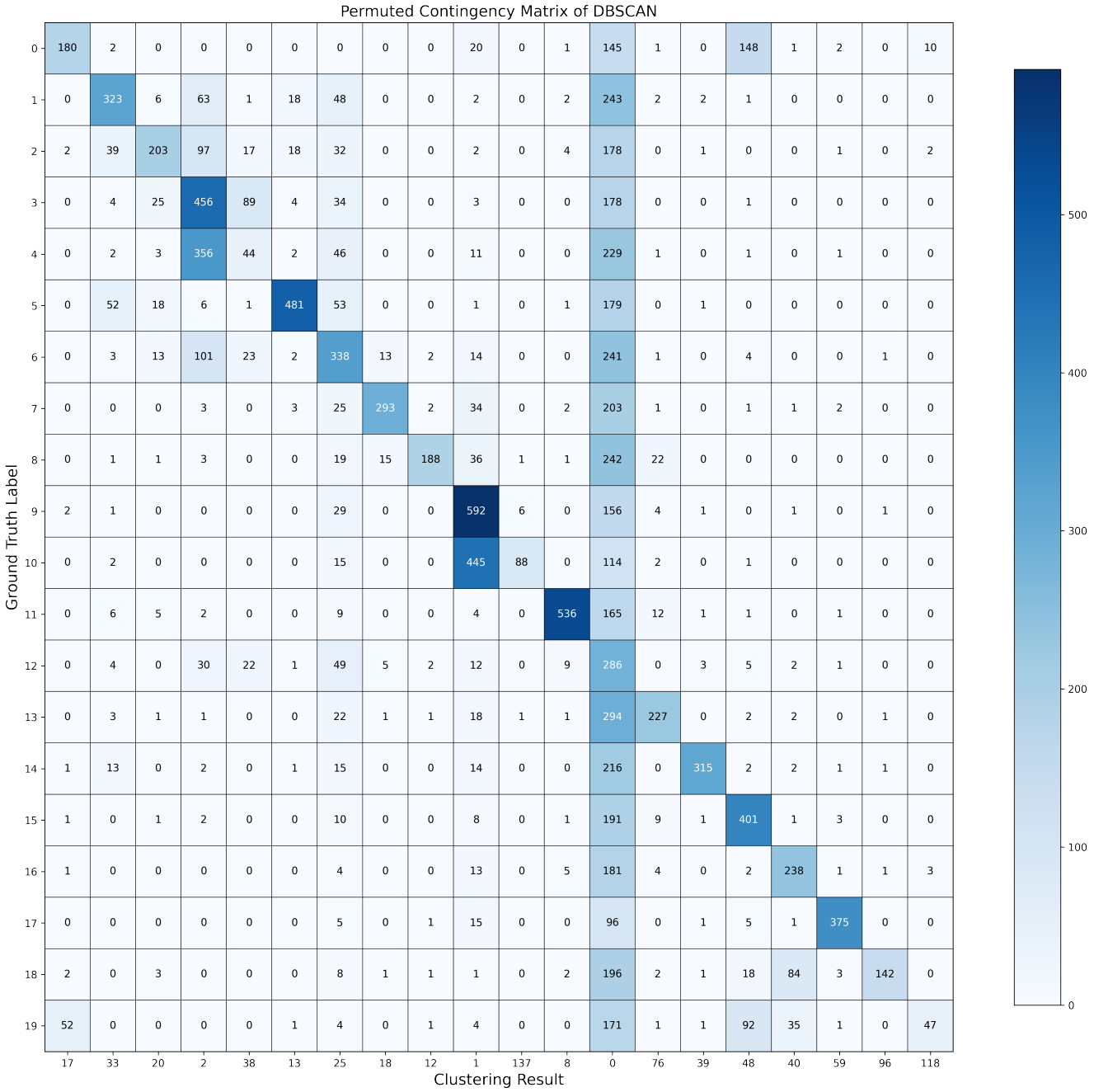

Figure 17: Contingency Matrix of DBSCAN (eps=0.2, min_samples=5)

Part 2 - BBCNews Dataset

QUESTION 16:

Data Acquiring

The dataset for this question was found on Kaggle for the BBC News Classification competition. The training dataset consists of 1490 articles in five categories: business, tech, politics, sport, and entertainment.

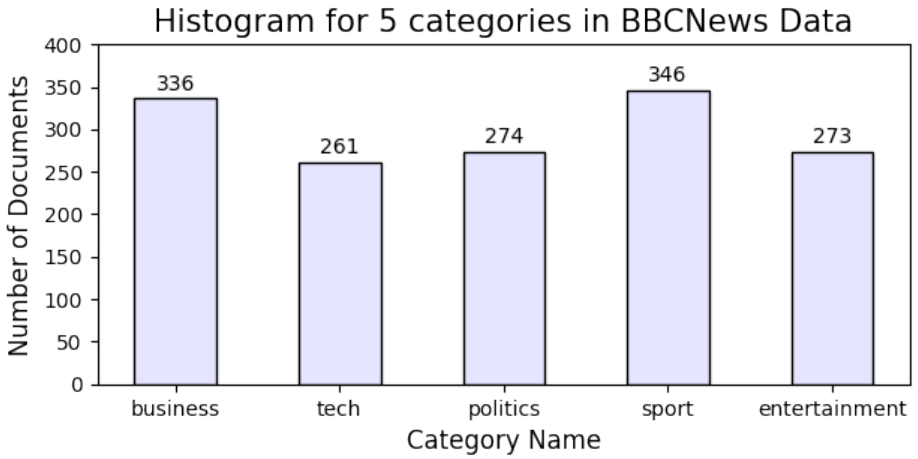

Figure 18: Histogram of training documents for each of 5 categories in BBCNews Dataset

We calculated the mean value and standard deviation to quantitatively evaluate whether the dataset is evenly distributed besides plotting the histogram in Table 18, and the result is as follows:

mean = 298 documents/category

std = 35.54

Since the raw data is already a text corpus, we directly transform the dataset to TF-IDF matrix using function "CountVectorizer" and "TfidfTransformer" as we did before, and the resulting dimensions of the TF-IDF matrix is: 1490×10557.

Feature Extraction

To determine the best feature extraction algorithm, we run experiments to look for the optimal number of components (parameter r) for different dimensionality reduction algorithms on the TF-IDF matrix, using the same method as in Question 5.

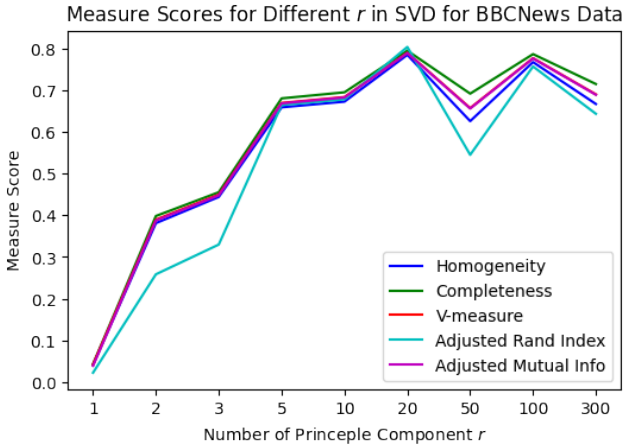

Figure 19: Performance of SVD vs r

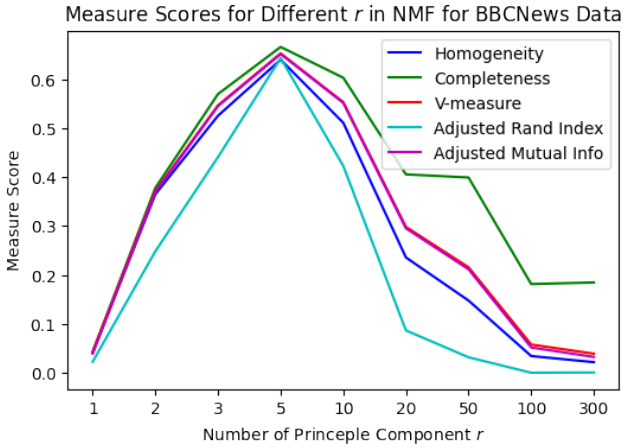

Figure 20: NMF with Frobenius norm vs r

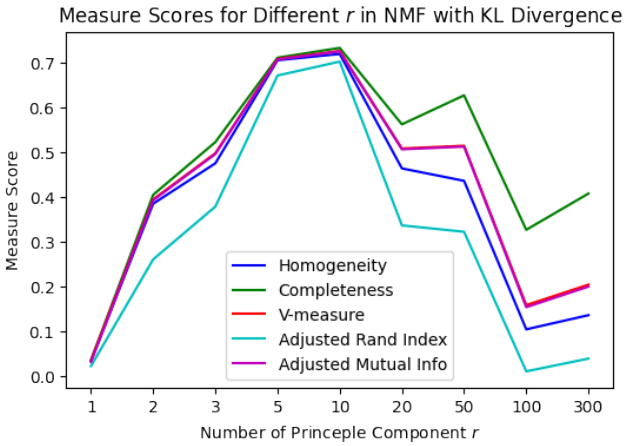

Figure 21: NMF with KL Divergence vs r

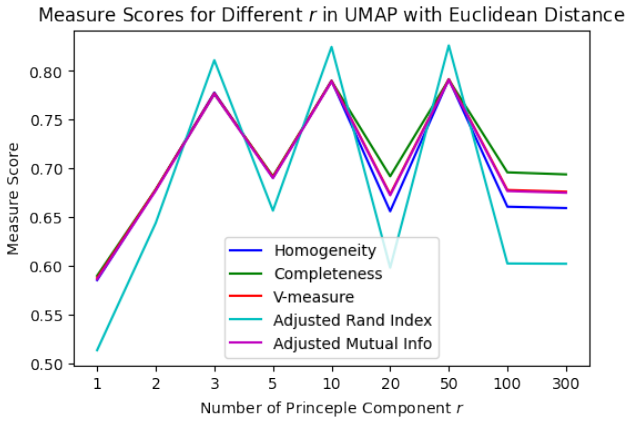

Figure 22: UMAP with Euclidian vs r

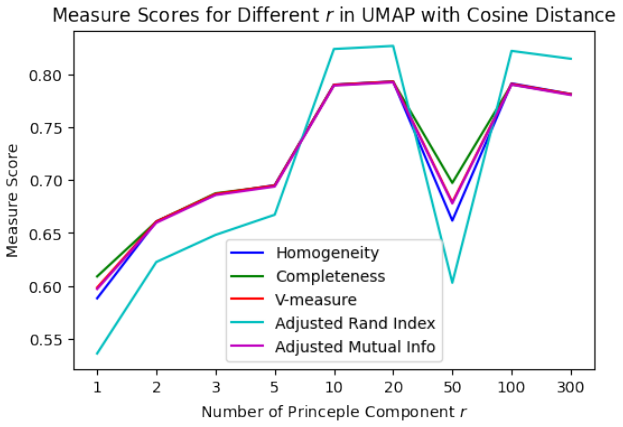

Figure 23: UMAP with Cosine vs r

We ran SVD, NMF with Frobenius norm, NMF with KL Divergence, UMAP with Euclidean distance, and UMAP with Cosine distance on the TF-IDF matrix over various number of components, respectively, as shown in Fig. 19-23. The optimal value of r and the best results for each of the algorithms are reported in Table 8.

Table 8: Evaluation results of Different Dimensionality Reduction Methods with best r.

Algorithm	r	Homogeneity	Completeness	V-measure	Adjusted Rand Index	Adjusted Mutual Info Score
SVD	20	0.7842	**0.7943**	0.7892	0.8032	0.7885
NMF (F Means)	5	0.6397	0.6657	0.6524	0.6430	0.6512
NMF (KL Means)	10	0.7190	0.7324	0.7257	0.7020	0.7247
UMAP (Euclidean)	50	0.7916	0.7911	0.7914	0.8262	0.7907
UMAP (Cosine)	20	**0.7937**	0.7934	**0.7936**	**0.8274**	**0.7929**

We can see that the best dimensionality reduction method is UMAP with Cosine distance and n_components=20.

Clustering

In order to perform clustering, we experimented on all five algorithms provided in the Part 1 of the project: KMeans Clustering, Agglomerative Clustering (with "ward" or "single" linkage criterion), DBSCAN and HDBSCAN.

For DBSCAN and HDBSCAN, We perform a grid search to find the optimal hyperparameter combination of *eps* and *min_samples* as in Question 14. We found that for DBSCAN, the best hyperparameter combination was eps = 0.5 and min_samples = 19 which resulted in 15 clusters. For HDBSCAN, we found that the best hyperparameter combination was min_samples = 45 and any value of eps which resulted in 3 clusters.

The evaluation results of all five clustering algorithms using the UMAP-reduced data and optimal hyperparameters are reported in Table 9.

Table 9: Evaluation results of Clustering Algorithms.

Algorithm	Homogeneity	Completeness	V-measure	Adjusted Rand Index	Adjusted Mutual Info Score
KMeans	**0.7923**	**0.7934**	**0.7936**	**0.8274**	**0.7929**
Agglomerative (ward)	0.7742	0.7753	0.7748	0.8074	0.7740
Agglomerative (single)	0.2358	0.5734	0.3343	0.1178	0.3308
DBSCAN	0.7043	0.4738	0.5665	0.3927	0.5623
HDBSCAN	0.3074	0.6378	0.4148	0.2358	0.4135

Performance Evaluation

From Table 9, we find that the best clustering algorithm for BBCNews dataset is KMeans, the visualization of the clustering result is shown in Fig. 24, and the contingency matrix of the clustering result is shown in Fig. 25.

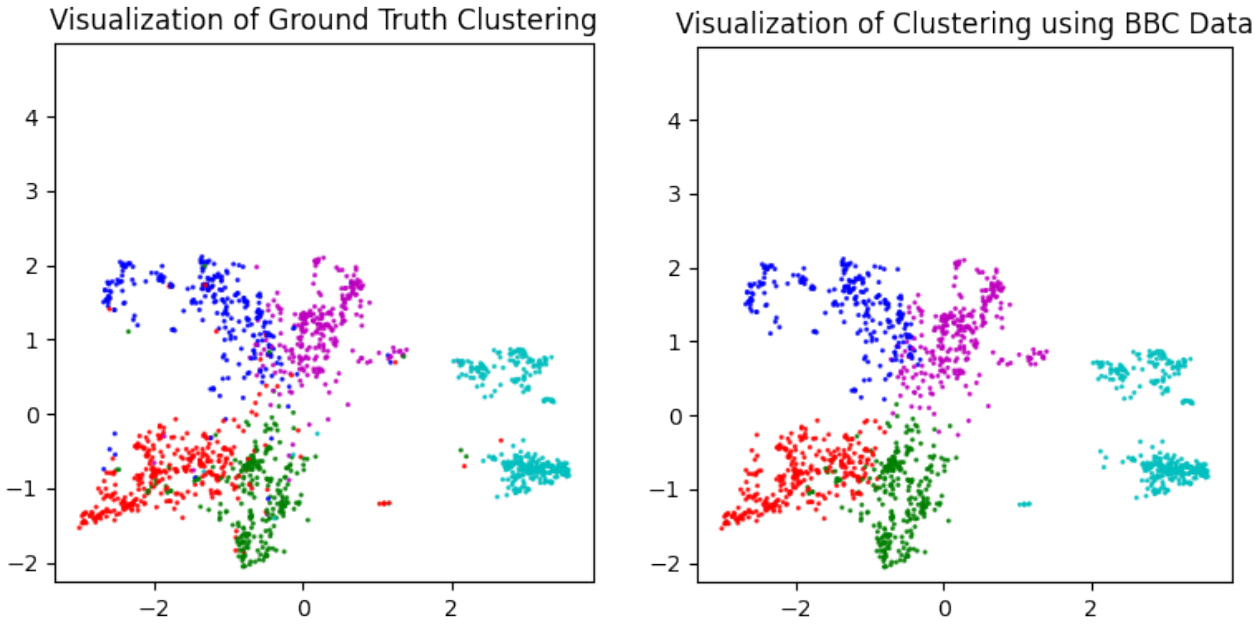

Figure 24: Visualization of the best clustering result for BBCNews Dataset.

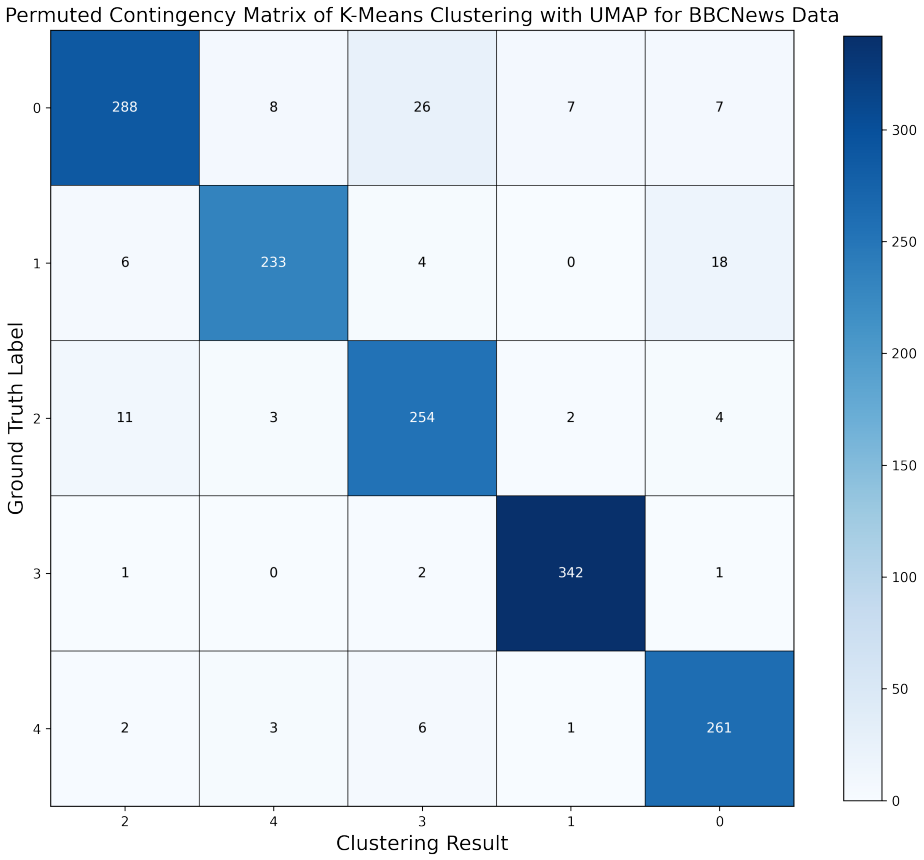

Figure 25: Contingency Matrix of KMeans Clustering with UMAP based on Cosine Distance.

The visualization of clustering result in Fig. 24 is permuted, so that the same color implies the corresponding clusters from the prediction to the ground truth, which is not the case for most of other visualizations in this report. Intuitively, we can see that the clustering result is very satisfactory since most of the data points are correctly clustered, which is also indicated by the contingency matrix in Fig. 25.

However, we can still find out a few misclassified data points, especially in the area that is near the edge of two K-Means cluster centers. But overall, we consider that the clustering performance is already pretty good.

Visualization of some other clustering algorithms

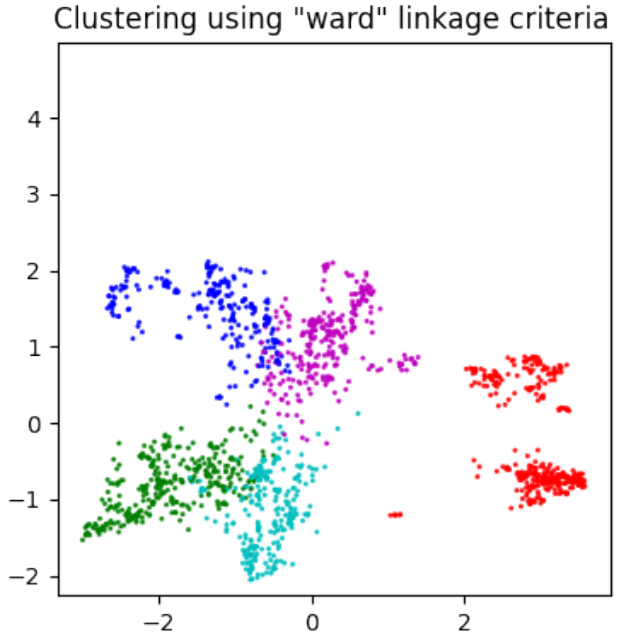

Figure 26: Agglomerative clustering result (ward) for BBCNews Dataset.

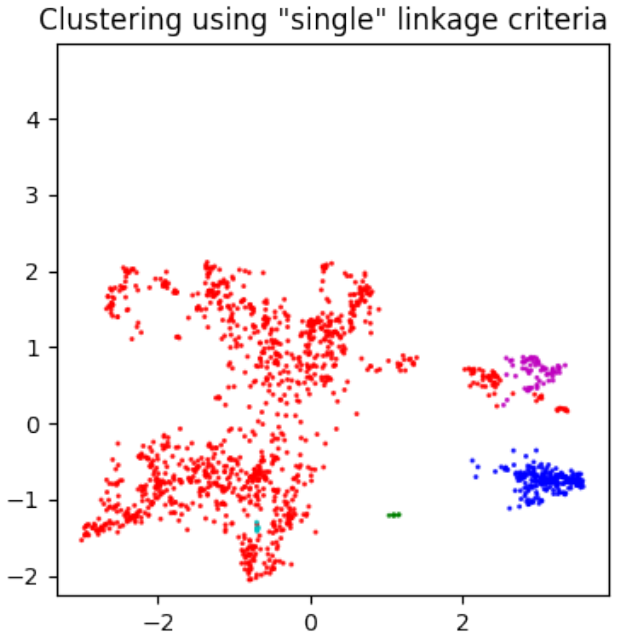

Figure 27: Agglomerative clustering result (single) for BBCNews Dataset.

Fig. 26 shows the clustering result for Agglomerative Clustering using the "ward" linkage criteria. Visually, the results are very similar to Fig. 24, which is expected, because its performance of five measures was only marginally lower than that of the K-Means clustering algorithm.

Fig. 27 shows the clustering results for Agglomerative clustering using the "single" linkage criteria. This algorithm performed much worse than Agglomerative clustering with the ward criteria. This is because that the single linkage criteria uses the minimum distance between two sets of points to determine if they should be merged. In the ground truth visualization, we can see that there are 4 distinct but very closely located clusters. Due to using the minimum distance criteria, these four clusters are not preserved.

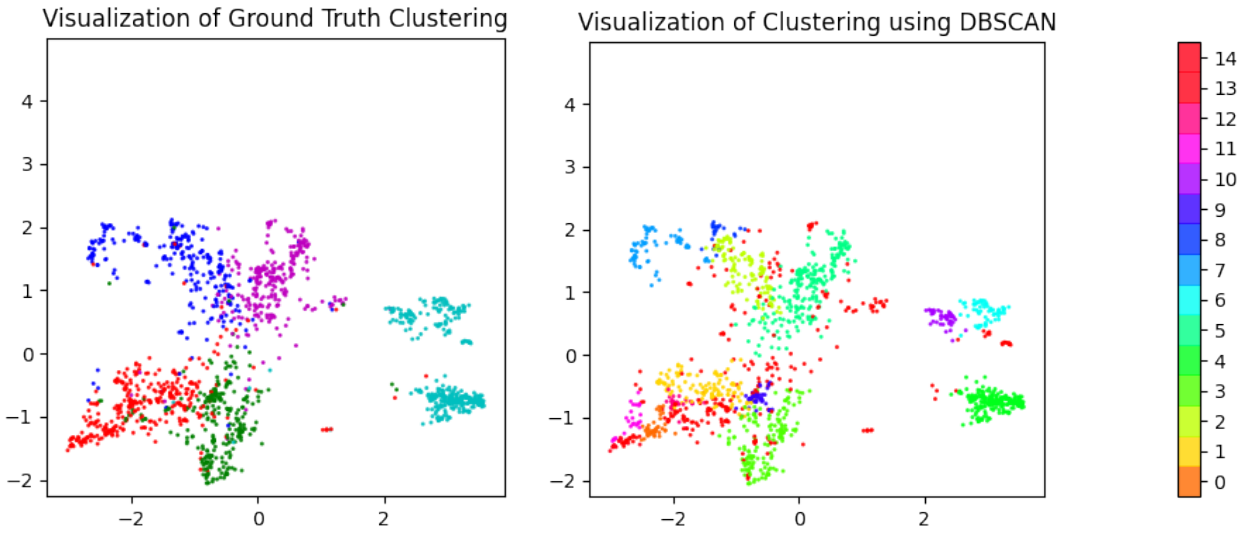

Figure 28: Visualization of DBSCAN clustering result for BBCNews Dataset.

Fig. 28 shows the clustering results for DBSCAN clustering. Under the optimal hyperparameters we discovered in the previous section, we found that DBSCAN gives 15 clusters for our data. However, we our data is actually divided into 5 classes which may mean that DBSCAN is finding clusters on a finer granularity than our data is actually labeled.

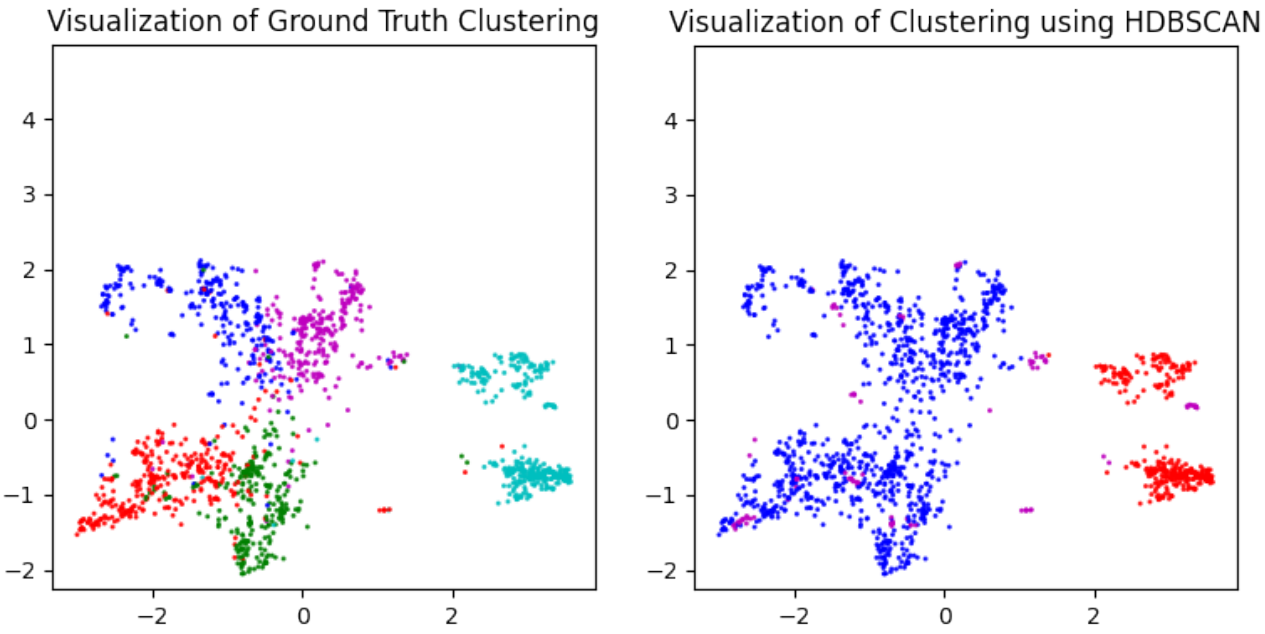

Figure 29: Visualization of HDBSCAN clustering result for BBCNews Dataset.

Fig. 29 shows the clustering results for HDBSCAN clustering. Similar to Agglomerative clustering with the single linkage criterion in Fig. 27, HDBSCAN has difficulty distinguishing between distinct but closely located clusters. This results in HDBSCAN returning 3 clusters with the optimal hyperparameters.

Reference

[1] Homogeniety_and_Completeness.pdf on CCLE

[2] BBC News Classification on Kaggle

[3] How do NMF and SVD differ - discuss.analyticsvidhya

[4] Euclidean distance and Cosine angle

[5] sklearn documentation of "AgglomerativeClustering"

[6] Algorithms for Nonnegative Matrix Factorization with the Kullback-Leibler Divergence

[7] sklearn documentation of "DBSCAN"