Project 2: Data Representations and Clustering

ECE 219 - Large Scale Data Mining
University of California, Los Angeles

Member 1:	Shruti Mohanty (705494615)
Member 2:	Kimaya Kulkarni (805528337)

1 Question 1

Report the dimensions of the TF-IDF matrix you obtain.

We are working with the "20 Newsgroups" dataset which is a collection of 20,000 documents that are approximately evenly partitioned across 20 different news groups. This is loaded using the scikit-learn library "fetch_20newsgroups". We start with a well-separated subset of samples from this larger dataset.

Class 1 has 'comp.graphics','comp.os.ms-windows.misc','comp.sys.ibm.pc.hardware', and 'comp.sys.mac.hardware'
Class 2 has 'rec.autos','rec.motorcycles','rec.sport.baseball', and'rec.sport.hockey'.

This question asks us to convert the data from these 8 categories into Term Frequency-Inverse Document Frequency (TF-IDF) matrix. The shape of the resulting matrix is **(7882, 23522)**.
The steps that were employed to obtain this TF-IDF matrix are -

- We removed headers and footers while importing the dataset using remove argument, and selected only the above mentioned categories. Further, the data was shuffled, and random state 0 has been used.
- Used CountVectorizer() to convert the text without any word normalization into bag of words matrix that contains frequencies of words in the vocabulary. The stopwords argument is set to "english" to remove words that provide no context, and min_df is set to 3.
- This matrix is converted into TF-IDF matrix using TfidfTransformer(). The TF-IDF matrix provides more useful and distinguishing features over bag-of-words for the classifier we want to use.

2 Question 2

Report the contingency table of your clustering result. You may use the provided plotmat.py to visualize the matrix. Does the contingency matrix have to be square-shaped?

Clustering is the process of finding groups of data that share similarities in the feature space without making use of any annotations.

K-means clustering is applied on the TF-IDF matrix obtained in the previous question, and it looks for centroid of K disjoint clusters of equal variance within the multimodal feature space of the dataset. The algorithm works by allocating a certain data point to a cluster such that the in-cluster inertia, which is the sum of squares, is minimized. The points in a cluster are expected to lie near the centroid.

K-means is an iterative algorithm which has the following steps repeated after the cluster centroids have been initialized randomly. -

- Each sample is assigned to the nearest centroid.
- New centroids are created by taking average of all the samples that are previously assigned to the old centroid cluster.

The stopping criteria for this algorithm depends on the number of iterations defined, or if centroids have become stable across iterations. We have to find the center of the cluster μ_k and r_{nk} such that the sum of squares is minimized, and x_n is the datapoint here.

$$\min J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} ||\mathrm{x}_n - \mu_k||^2$$

$$[r_{nk} = \begin{cases} 1, & \text{if } \mathrm{x}_n \text{ is assigned to cluster } k \\ 0, & \text{otherwise.} \end{cases}]$$

The covariance matrix obtained is diagonal and all equal. The contingency table is a generalization of the confusion matrix, where the rows show the number of samples in ground truth clusters, and the columns show the number of samples in ground truth clusters assigned by the K-means algorithm.

The below figure shows the contingency matrix for k-means clustering on the TF-IDF matrix for the subset of data for computer technology and recreational activity. The sklearn.cluster.KMeans library is used for K_means clustering, The n_clusters is set to 2, and the initialisation used is k_means++. k-means++ selects initial cluster centers for k-mean clustering in a smart way to speed up convergence. Max_iter is set to 1000, and n_init is 30 (it indicates the number of times the algorithm reinitializes the centroid seed in k_means++), and the random state is set to 0. We can observe that the clustering algorithm did a good job correctly assigning most of the sample points to their respective clusters with the selected hyperparameters as we had 2 classes and 2 clusters have been formed by our algorithm with most data belonging to them.

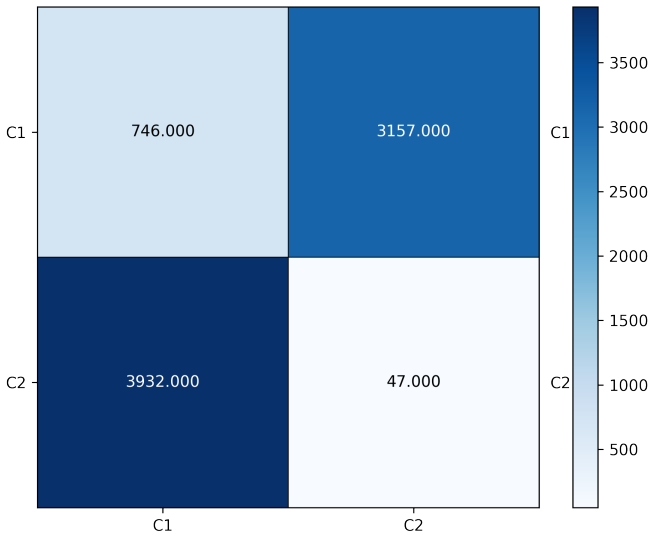

Figure 1:
Contingency matrix for K=2 clusters on TF-IDF matrix

The contingency matrix need not always be square shaped. In this example, we knew the number of target classes, and the number of clusters was chosen accordingly, hence it was square shaped. But if we weren't aware of the target classes, then we could also obtain rectangular contingency matrices based on the number of cluster parameter we choose, for example if we make k=3.

3 Question 3

Report the 5 clustering measures explained in the introduction for Kmeans clustering.

The metrics we are asked to report in this question are -

- Homogeneity - It is a measure of how "pure" the clusters are. If each cluster contains only data points from a single class, the homogeneity is satisfied. Homogenity (h) is independent of absolute value of ground truth values and it depends on conditional entropy of labels C given cluster assignments K.

$$h = 1 - \frac{H(C|K)}{H(C)}$$

$$H(C|K) = -\sum_{c=1}^{|C|}\sum_{k=1}^{|K|}\frac{n_{c,k}}{n}\cdot\log\left(\frac{n_{c,k}}{n_k}\right)$$

$$H(X) = -\sum_{h=1}^{|H|}\frac{n_h}{n}\cdot\log\left(\frac{n_h}{n}\right)$$

$$n_{c,k} = n_k \cap n_c$$

H(X) denotes the entropy of partition X, where X denotes non-overlapping groups of sample points.The homogeneity score is maximized when each cluster K_i contains samples only from the class C_i.

- Completeness - It indicates how much of the data points of a class are assigned to the same cluster.Completeness is defined in terms of conditional entropy of clusters K given ground truth labels C . It is the maximum when each ground truth class C_i is part of a cluster K_i.

- V-Measure - It is the harmonic average of homogeneity score and completeness score. Compared to homogeneity and completeness, this metric is symmetric and is useful for measuring agreement between independent cluster assignment strategies on the same dataset.

- Adjusted Rand Index - This metric is similar to accuracy and it computes the similarity between the clustering labels and the ground truth labels. It takes into account all the pairs of points that fall either in the same cluster and the same class or in different clusters and different classes.

$$\text{ARI} = \frac{\text{RI} - \mathbf{E}(\text{RI})}{\max(\text{RI}) - \mathbf{E}(\text{RI})}$$

$$\text{RI} = \frac{TP + TN}{TP + FP + TN + FN}$$

- Adjusted mutual information score measures the mutual information between the cluster label distribution and the ground truth label distributions, that are adjusted for chance. In the equation MI(C,K) measures the dependency of clustering labels on ground truth labels and vice versa. -

$$\text{AMI} = \frac{\text{MI(C,K)} - \mathbf{E}(\text{MI(C,K)})}{\text{avg(H(C), H(V))} - \mathbf{E}(\text{MI(C,K)})}$$

The measures for these metrics obtained by us are -

- Homogeneity: 0.579
- Completeness: 0.594
- V-measure: 0.587
- Adjusted Rand-Index (ARI): 0.638
- Adjusted Mutual Information Score (AMI): 0.587

From these results we observe that the homogeneity is approximately 60%, which indicates that 40% of the samples in a particular cluster are not from the assigned ground truth label. The completeness is also approximately 60%, which indicates that 40% of the samples for a given class label are not part of the same cluster. The ARI and AMI scores indicate that the clustering distribution is similar to the ground truth label distribution to moderate extent (nearly 63% and 59% respectively).

4 Question 4

Report the plot of the percentage of variance that the top r principle components retain v.s. r, for r = 1 to 1000.

As dimensions of the data increases, the data starts becoming sparse in each dimension because of the rapid increase in volume. This is commonly referred to as the Curse of Dimensionality. This causes the euclidean distances to almost be the same, inturn degrading the clustering performance. In this part we try to find a "better" representation tailored to the performance of the downstream task of K-means clustering. Towards finding a better representation, we can reduce the dimension of our data with different methods before clustering. We try out Singular Value Decomposition (SVD) or Latent Semantic Indexing (LSI) for dimensionality reduction.

LSI or SVD reduces the rank of the feature matrix by multiplying the TF-IDF matrix calculated in the previous question with the first k principal components in the feature space using fit_transform function. It is represented in columns of matrix V_k , which is found using singular value decomposition (SVD) of the TF-IDF matrix: The goal is to reduce the Frobenius norm of the difference between X and $U_k \sum V_k^T$.

$$\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T$$
$$\mathbf{X}_{\text{reduced}} = \mathbf{X}\mathbf{V}_k$$

k was chosen as 1000 as indicated in the question.

To obtain the SVD of the TF-IDF matrix, we used the TruncatedSVD() function with n_components set to 1000 and random_state set to 0. We use fit and transform on the dataset features from TF-IDF matrix. To calculate the variance retention percentage of top r principal components, we used the cumulative sum of the explained_variance_ratio . The figure below shows the plot of the percentage of variance that the top r principle components retain v.s. r, for r = 1 to 1000. We can see that as r increases, the variance retention percentage is also uniformly increasing. It goes as high as close to 60%.

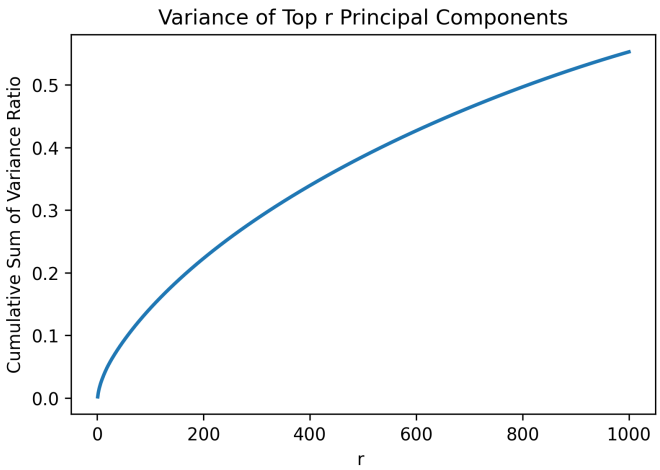

Figure 2:
Plot of the percentage of variance that the top r principle components retain v.s. r, for r = 1 to 1000

5 Question 5

Try r = 1, 2, 3, 5, 10, 20, 50, 100, 300, and plot the 5 measure scores v.s. r for both SVD and NMF. Report a good choice of r for SVD and NMF respectively.

We are asked to reduce the dimensionality of the TF-IDF matrix to obtain dense text representations, using both SVD and Non-negative Matrix Factorization (NMF), with the goal of obtaining the ideal value of the number of principal components r in terms of the metrics mentioned in this report.

NMF attempts to find two non-negative matrices W and H whose product is as close to X as possible, with W being the low-rank feature matrix for X, where r = number of topics. H (coefficient matrix) provides weighing factors for all the topics in each document, while W (basis vectors) corresponds to the clusters discovered in the document. The solutions for W and H are found by solving this optimization problem, where they are constrained to be non-negative -

$$\min_{\{\mathbf{W},\mathbf{H}\}\geq 0} ||\mathbf{X} - \mathbf{WH}||_F^2$$

We used the NMF() function to obtain W. The parameters for NMF were set as init='random', random_state=0, max_iter=200. The possible values of r, i.e n_components included 1, 2, 3, 5, 10, 20, 50, 100 and 300. Figures 3, 4, 5, 6 and 7 show the plots of the 5 metrics vs r for both SVD and NMF for the above mentioned values of r , homogeneity, completeness, V-measure, ARI and AMI respectively. Figure 8 has the mean of all 5 metrics for SVD, and NMF. We perform our analysis for the best r value based on these plots.

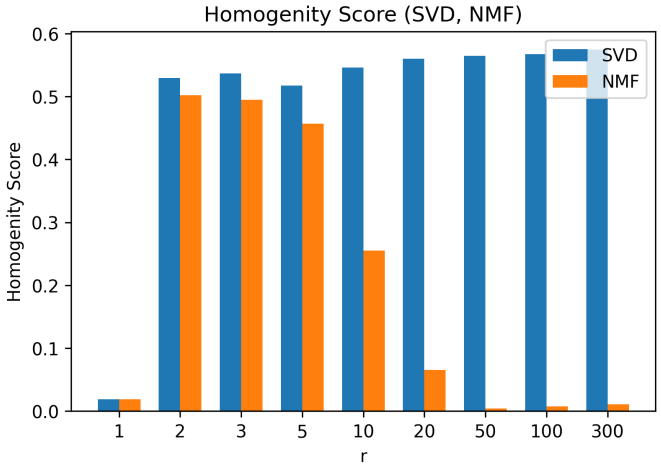

Figure 3:
Plot of Homogenity Score v/s r

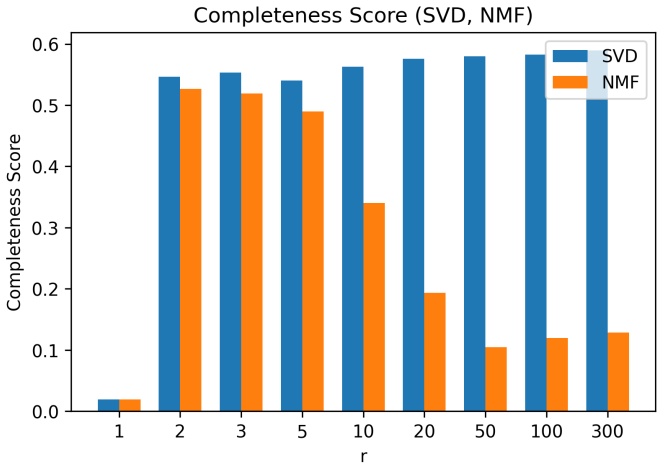

Figure 4:
Plot of Completeness Score v/s r

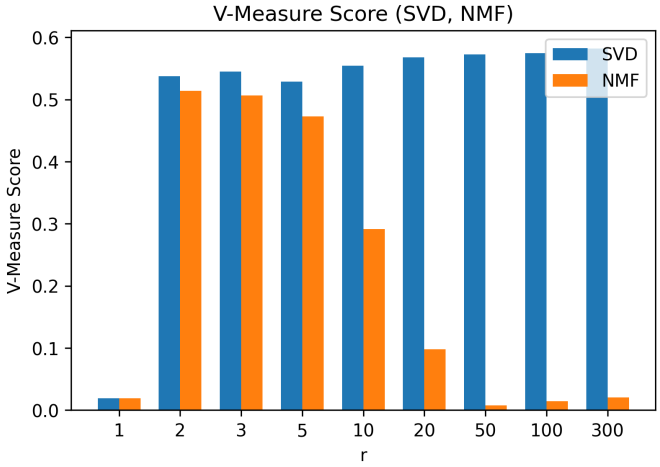

Figure 5:
Plot of V-Measure Score v/s r

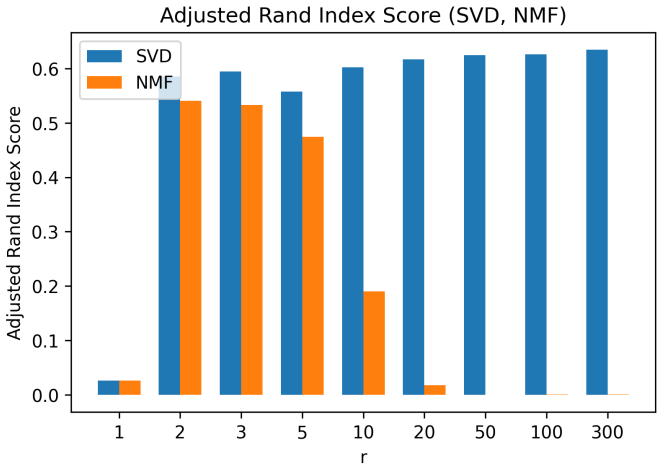

Figure 6:
Plot of Adjusted Rand Index Score v/s r

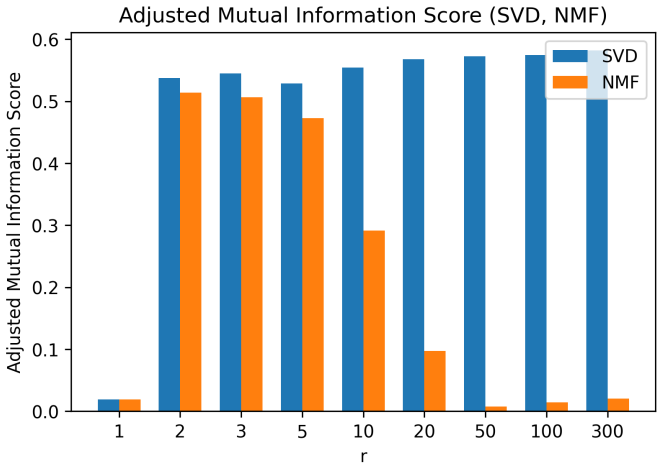

Figure 7:
Plot of Adjusted Mutual Information Score v/s r

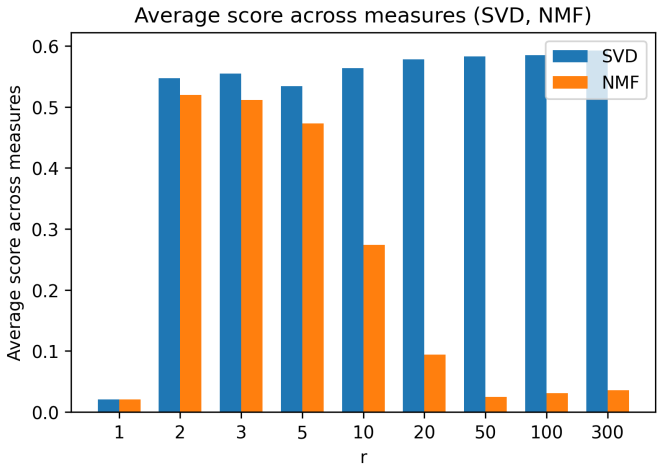

Figure 8:
Plot of Average Scores v/s r

We observe that the relationship between clustering metric and number of principal components is non-monotonic. We choose the best value of r after performing analysis on the plot in Figure 8 which is the average score across all the measures.

From the above Figure 8 we tend to choose the best value of r as 300 for SVD, and 2 for NMF. But, we can take into account the hint given in the question about the choice of r, there is a trade-off between the information preservation, and better performance of k-means in lower dimensions. When r increases, the clustering performance drops as Euclidean Distances converges to a same value between the equidistant features. And this is a key metric for the performance of K-means clustering. So, we need to choose a value of r that can give enough information, without affecting the performance of K-means clustering. From the plot, this value of r seems to be at 20 from SVD, as the performance doesn't increase significantly after that till 300.

So, overall from this analysis we choose r=20 for SVD, and r=2 for NMF.

6 Question 6

How do you explain the non-monotonic behavior of the measures as r increases?

From the previous question graphs we can conclude that there is no monotonic trend in the scores as r increases. We expect a larger value of r to preserve more information, and perform better. However, from our previous analysis we notice that as r increases the data gets more sparse, and leads to a noisy-feature matrix that degrades the performance of K-means clustering. This is because the Euclidean distances converges to a constant value between the equidistant sample points. The clustering algorithm can't find centroids with sufficient distance between them as the inertia is not normalized, and this leads to poor clustering performance.

This trend can very clearly be seen in NMF, where after r=2,3 the average values start dropping. In SVD, a significant drop isn't seen but the values are almost stagnant and saturated at a very low r value.

NMF allows only positive entries in the reduced rank feature matrix, whereas SVD has no restriction like this, because of which the results of SVD are more predictable compared to NMF. SVD is able to better represent higher-dimensional feature matrix providing a lesser information loss when compared to NMF. SVD is also more deterministic than NMF where there is a geometric basis ordered by relevance. This is very important in the high dimensional feature space for clustering. NMF does not consider geometry in the feature space basis. Since SVD produces a feature matrix with the most relevant features higher in the hierarchy, increasing the value of r does not cause significant drop or rise in clustering performance. From the K-means perspective, a feature matrix produced by SVD with higher values of r is not adding any significant distinguishable information. Also, unlike NMF there is no information loss, and the clustering performance is nearly constant because of the order of the basis and the euclidean distance. SVD remains constant through 20 to 200, however NMF drops after 2. SVD results are unique whereas, NMF is a stochastic algorithm with non-unique convergence results.

7 Question 7

Are these measures on average better than those computed in Question 3?

Measures computed in Question 3 (Without any feature reduction - sparse representation) -

- Homogeneity: 0.579
- Completeness: 0.594
- V-measure: 0.587
- Adjusted Rand-Index (ARI): 0.638
- Adjusted Mutual Information Score (AMI): 0.587

Measures computed from highest parameter of SVD after feature reduction (n_components=300)-

- Homogenity: 0.574
- Completeness: 0.589
- V-Measure: 0.581
- Adjusted Rand-Index (ARI): 0.634
- Adjusted Mutual Information Score (AMI): 0.581

Measures computed from best parameters of NMF after feature reduction (n_components=2)-

- Homogenity: 0.502
- Completeness: 0.526
- V Measure: 0.514
- Adjusted Rand-Index (ARI): 0.540
- Adjusted Mutual Information Score (AMI): 0.513

After obtaining a dense representation of the data (via. SVD and NMF), we compare the average metrics with sparse representation of data fed into K-means clustering. It is noticed that the data after reduction via SVD peforms almost similar to the sparse data when n_components chosen is 300. All the 5 metrics are very similar with the ARI being around 63%. NMF on the other hand doesn't perform comparably well like SVD when it's best parameter model is chosen. It's ARI is around 54% when n_components is set at 2. There are a lot of reasons for this trend, that has already been explained in the previous question and is again summarised here.

- In NMF, where after r=2,3 the average values start dropping. In SVD, a significant drop isn't seen but the values are almost stagnant and saturated at a very low r value from 20 to 300.
- NMF allows only positive entries in the reduced rank feature matrix, whereas SVD has no restriction like this, because of which the results of SVD are more predictable compared to NMF. SVD is able to better represent higher-dimensional feature matrix providing a lesser information loss when compared to NMF. SVD is also more deterministic than NMF where there is a geometric basis ordered by relevance. This is very important in the high dimensional feature space for clustering. NMF does not consider geometry in the feature space basis. Since SVD produces a feature matrix with the most relevant features higher in the hierarchy, increasing the value of r does not cause significant drop or rise in clustering performance.

- From the K-means perspective, a feature matrix produced by SVD with higher values of r is not adding any significant distinguishable information. Also, unlike NMF there is no information loss, and the clustering performance is nearly constant because of the order of the basis and the euclidean distance. SVD results are unique whereas, NMF is a stochastic algorithm with non-unique convergence results.

This explains why the results of SVD are comparable to the metrics we obtained in Question 3, but NMF performs worse.

8 Question 8

Visualize clustering results

In this question, we are asked to visualize the clustered samples points compared to ground truth labels for SVD and NMF on 2D-plane, with the best choices of r that is 20 for SVD and 2 for NMF. Figure 9,10 shows the plot for SVD with ground truth labels and k-means labels while Figure 11,12 shows the plot for NMF with ground truth labels and k-means labels.

The data in its decreasing order of components is mapped to 2D plane after feature reduction.

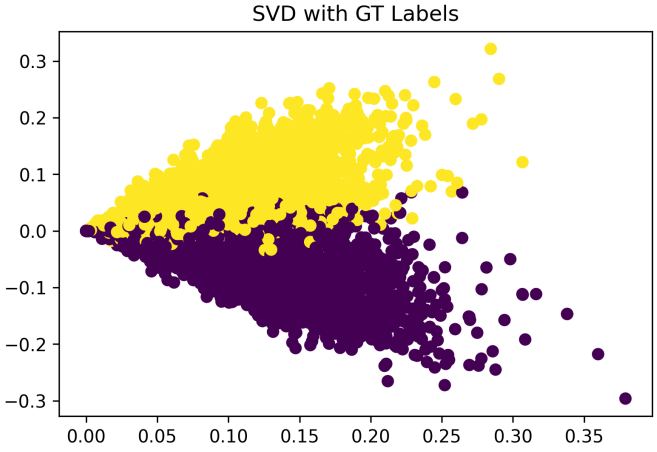

Figure 9:
SVD with GT labels for r=20

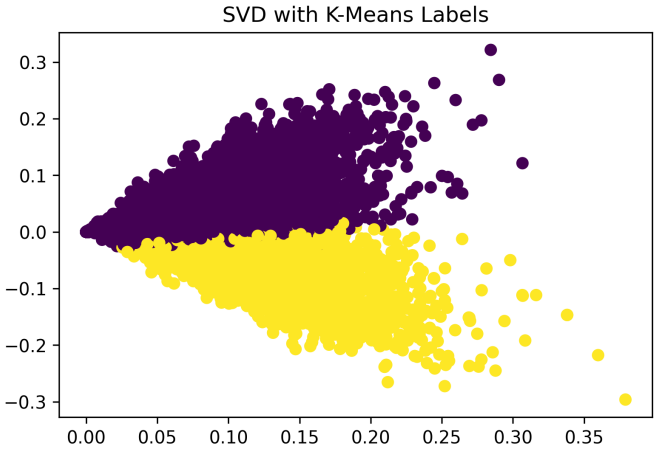

Figure 10:
SVD with K-means labels for r=20

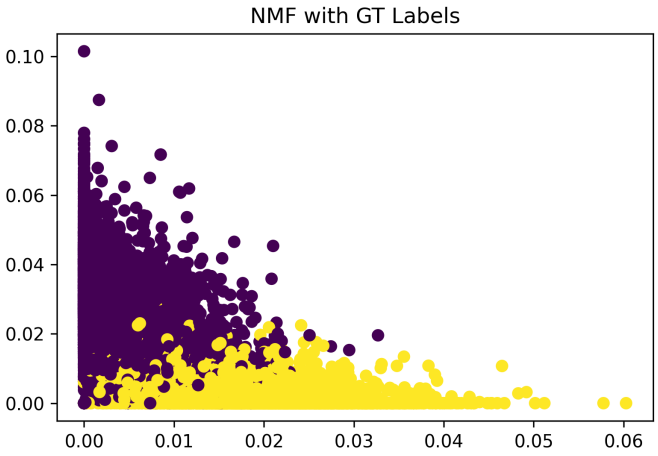

Figure 11:
NMF with GT labels for r=2

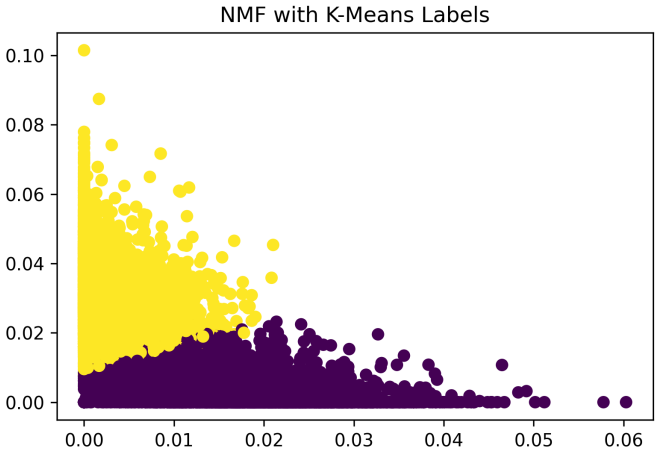

Figure 12:
NMF with k-means labels for r=2

9 Question 9

What do you observe in the visualization? How are the data points of the two classes distributed? Is distribution of the data ideal for K-Means clustering?

From the visualisation we can infer that this distribution of data is not ideal for k-means clustering, and it is not a univariant distribution.

- K-means clustering is under the assumption that the clusters are isotropic and convex. However, for both SVD and NMF, we don't see spherical distributions on the 2D plot. The plots are irregularly shaped and elongated blobs. For NMF, the blobs are unevenly sized across the 2 classes.
- K-means++ initializes the centroids far from each other. But, in this distribution we observe that there is a lot of overlap amongst the 2 clusters, as they are closely packed to one another. The Euclidean distance between the centroids of the 2 clusters is at the minimum, and this leads to no clear decision boundary between the 2 clusters. This is responsible for the low homogenity and v-measure scores. And for k-means, euclidean distance is an important parameter that affects clustering.
- Another assumption of K-means is that the clusters are univariant and gaussian in nature. However, from the scatter plots we can see that SVD and NMF have unequal variance for the two clusters, where we observe that the cluster along the row axis is much more compactly packed. These outliers and noise affect the centroids in the data.

10 Question 10

Visualize the contingency matrix and report the five clustering metrics for ALL 20 categories.

In this question we perform clustering for the entire 20 categories in the 20newsgroups dataset. The steps followed are similar to Question 1, but all categories are loaded here.

- We removed headers and footers while importing the dataset using remove argument. Further, the data was shuffled, and random state 0 has been used.
- Used CountVectorizer() to convert the text without any word normalization into bag of words matrix that contains frequencies of words in the vocabulary. The stopwords argument is set to "english" to remove words that provide no context, and min_df is set to 3.
- This matrix is converted into TF-IDF matrix using TfidfTransformer(). The TF-IDF matrix provides more useful and distinguishing features over bag-of-words for the classifier we want to use.
- The sparse data was reduced to a dense representation with SVD. A hyper parameter search was performed across the r values - 1,2,3,5,10,20,50,100 and 300.
- K-means clustering was performed with n_clusters set to 20. The other parameters set were init='k-means++', max_iter=1000, n_init=30 and random_state=0.

From the table , we can see that the best average metric was obtained for r = 100 at 0.312. As a result, for this question, we plot the contingency matrix and report the 5 clustering measures for SVD with r = 100. Sometimes due to mismatch between absolute labels and cluster labels, the high-value entries of the 20 × 20 contingency matrix can be scattered around, making it messy to inspect, even if the clustering result is not bad. To mitigate mismatch between absolute labels and cluster labels, we used scipy.optimize.linear_sum_assignment to identify the best-matching cluster-class pairs, and permute the columns of the contingency matrix accordingly.

Table 1: SVD with different r values with K means cluster =20

	r=1	r=2	r=3	r=5	r=10	r=20	r=50	r=100	r=300
Homogenity	0.0242	0.212	0.247	0.320	0.324	0.333	0.283	0.330	0.297
Completeness	0.026	0.224	0.265	0.348	0.354	0.375	0.355	0.399	0.372
V-Measure	0.025	0.218	0.255	0.333	0.338	0.353	0.315	0.362	0.330
ARI	0.005	0.065	0.083	0.125	0.122	0.120	0.079	0.111	0.087
AMI	0.022	0.215	0.253	0.331	0.336	0.350	0.313	0.359	0.328

The best value of r from the above table is 100. The clustering metrics reported on that are -

- Homogeneity: 0.330
- Completeness: 0.399
- V-measure: 0.362
- Adjusted Rand-Index (ARI): 0.111
- Adjusted Mutual Information Score (AMI): 0.359

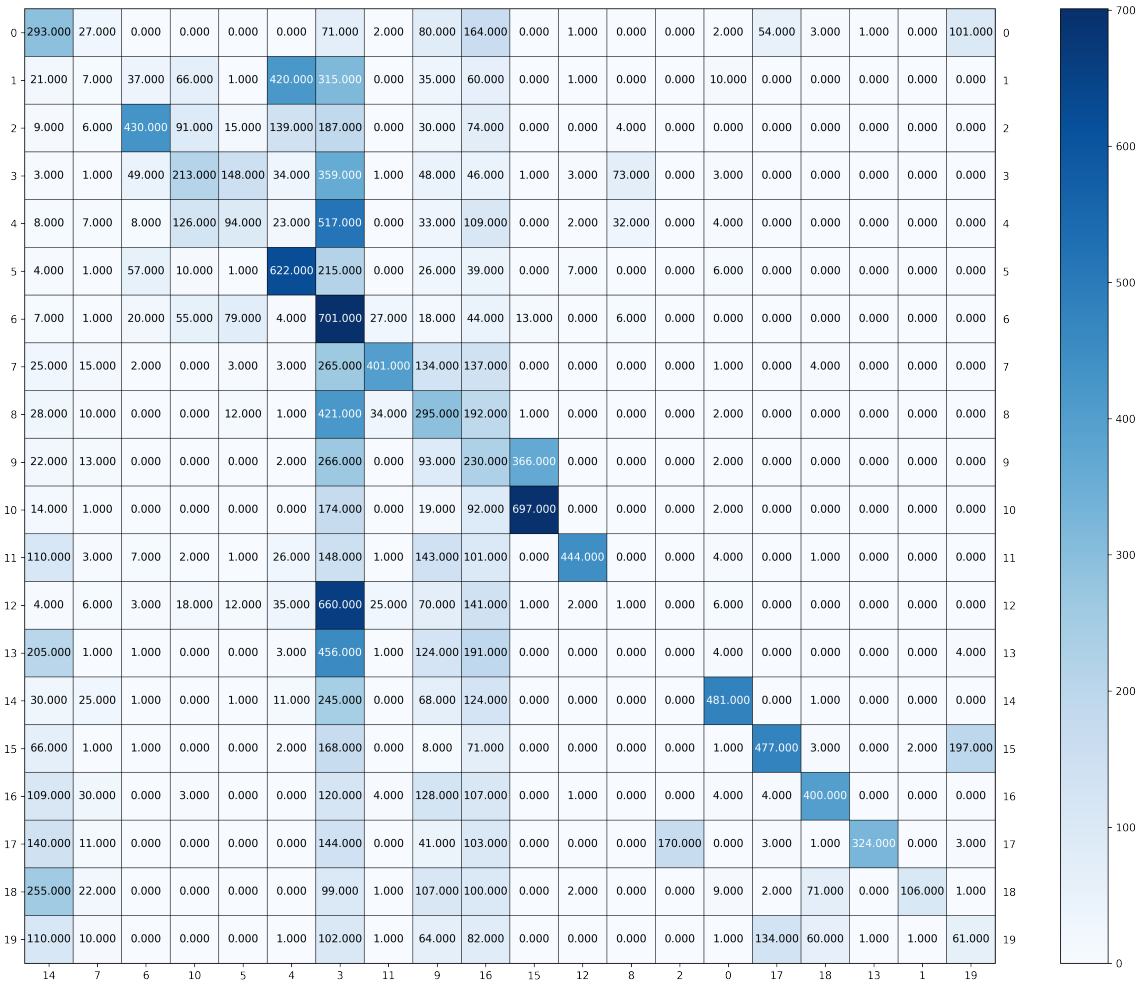

Figure 13:
Contingency matrix with SVD r= 100, and K-means cluster= 20

11 Question 11

Report the permuted contingency matrix and the five clustering evaluation metrics for "euclidean" and "cosine" for UMAP.

For this question, we are asked to use Uniform Manifold Approximation and Projection (UMAP) for dimension reduction and find the ideal number of principal components for two distance metrics, euclidean and cosine distance. The motivation behind using UMAP is - consider two documents that are about the same topic and are similar, but one is very long while the other is short. The magnitude of the TF-IDF vector will be high for the long document and low for the short one, even though the orientation of their TF-IDF vectors might be very close. In this case, the cosine distance adopted by UMAP will correctly identify the similarity, whereas Euclidean distance might fail. To verify these characteristics we experiment with both the distance metrics.

UMAP constructs graphical representation of the data in high-dimensions and attempts to optimize a low-dimensional graphical embedding to be as geometrically similar as possible to the high-dimensional graph. The high-dimensional graph is constructed using a weighted graph with the edges representing probability of association of two samples. The association radius is locally determined via the distance between k nearest neighbors to a sample point, with decreasing connection likelihood as the radius increases. The number of neighbours controls how UMAP balances local versus global structure.. UMAP then projects the data into lower dimensions using a directed graph layout algorithm, with a hyperparameter controlling how tightly or loosely lumped the embedding points are going to be.

The parameters with which we conduct our analysis for UMAP were selected from the table given in Question 17, where UMAP can take n_components = 5,20,200. The table below reports the values for those parameters for euclidean and cosine distances.

Table 2: UMAP (Euclidean) with different r values with K means cluster= 20

	r=5	r=20	r=200
Homogenity	0.007	0.006	0.008
Completeness	0.007	0.006	0.008
V-Measure	0.007	0.006	0.008
ARI	0.001	0.001	0.001
AMI	0.004	0.003	0.005

Table 3: UMAP (Cosine) with different r values with K means cluster= 20

	r=5	r=20	r=200
Homogenity	0.571	0.571	0.563
Completeness	0.596	0.596	0.598
V-Measure	0.583	0.583	0.580
ARI	0.448	0.448	0.443
AMI	0.582	0.582	0.579

From the table, on comparing the average metrics across these r values we obtain the best value

of r for UMAP (euclidean) as 200, and UMAP(cosine) as 5. The reported clustering metrics for these values are -

UMAP Euclidean :

- Homogeneity: 0.008
- Completeness: 0.008
- V-measure: 0.008
- Adjusted Rand-Index (ARI): 0.001
- Adjusted Mutual Information Score (AMI): 0.005

UMAP Cosine -

- Homogeneity: 0.571
- Completeness: 0.596
- V-measure: 0.583
- Adjusted Rand-Index (ARI): 0.448
- Adjusted Mutual Information Score (AMI): 0.582

From the metrics, we can see that UMAP with cosine distance metric performs much better clustering than UMAP with Euclidean distance with ARI approximately at 45% compared to 0.1%. This is expected behaviour as per the hint given in the question. Cosine similarity is not affected by the magnitude of the vectors, meaning that the length of the documents does not affect the distance metric, but rather it associates clusters based on angle between sample points. This helps correct the fact that higher frequency of words in a document does not necessarily mean it belongs to a certain class given those words, it could also mean the document is very long. In addition, in high dimensions, the rapid increase in volume causes the data to become sparse in each dimension, causing the Euclidean distances to converge to a constant value between all sample points. In K-means this posed as a problem for us where the euclidean distance became similar for sparse data. This is overcome with UMAP cosine. Since all feature points become equidistant, UMAP euclidean cannot find distinguishable clusters within the data. Thus, for textual clustering, cosine similarity is the ideal metric over L2 distances. The contingency matrix for Euclidean UMAP (r = 200) and Cosine UMAP (r = 5) are shown in the below figures.

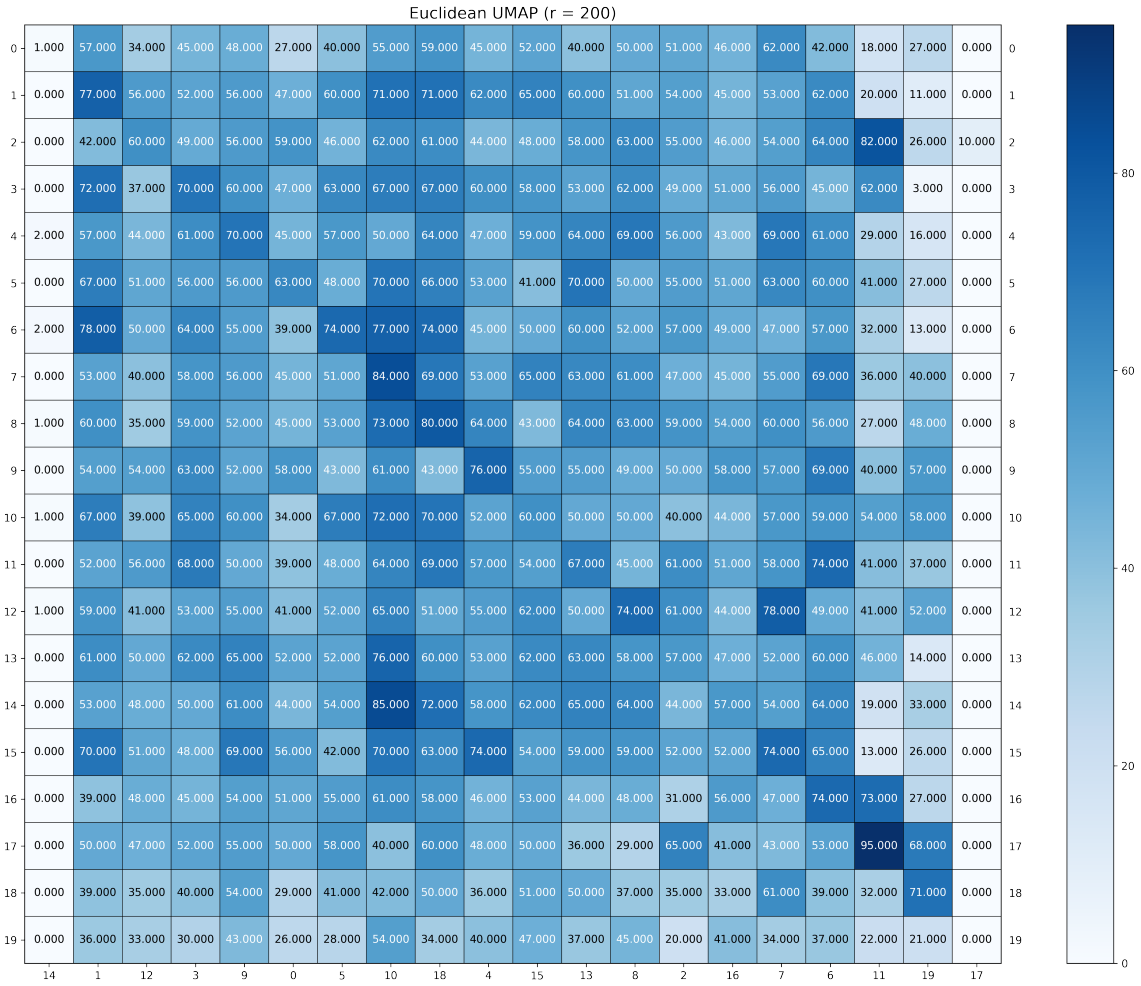

Figure 14:
Contingency matrix with UMAP (euclidean) r= 200, and K-means cluster= 20

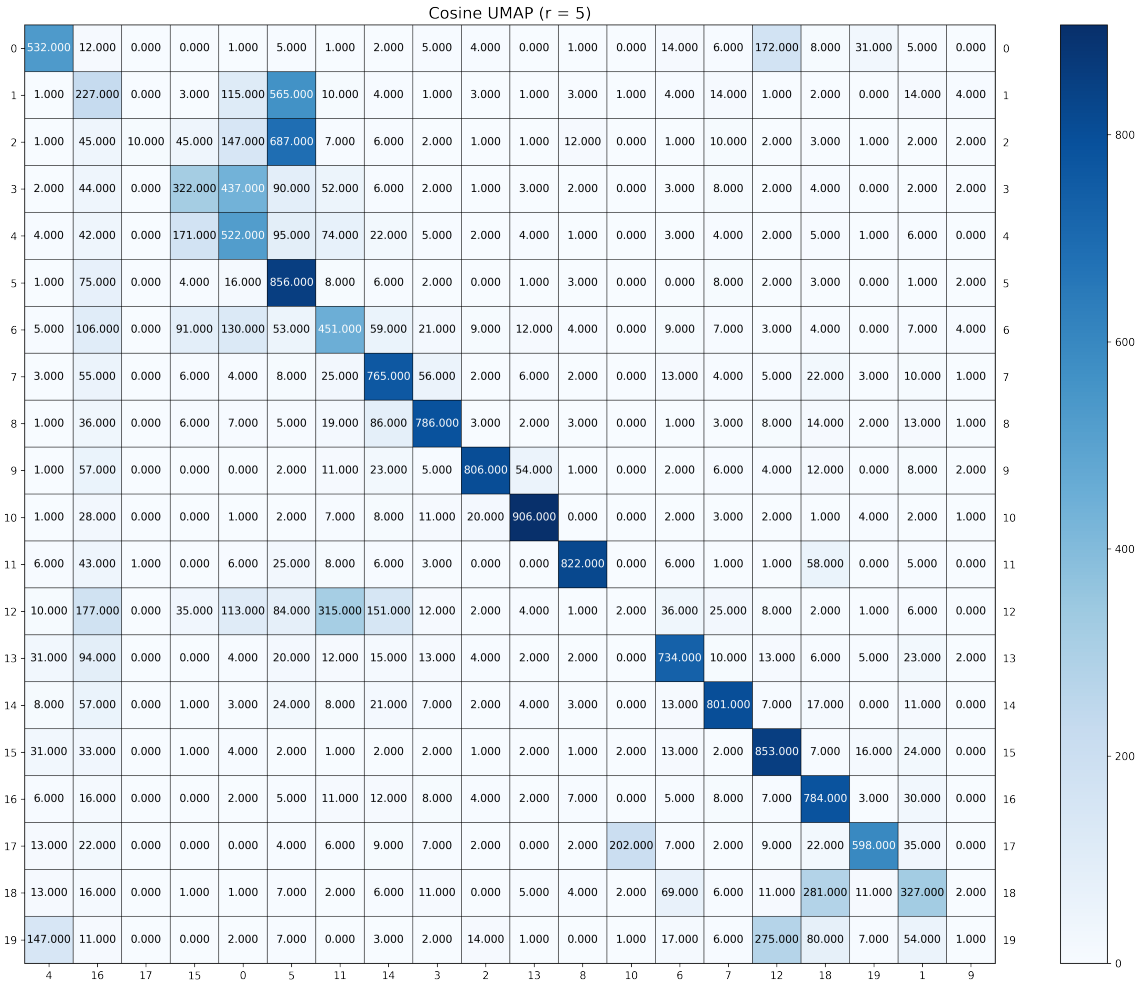

Figure 15:
Contingency matrix with UMAP (cosine) r= 5, and K-means cluster= 20

12 Question 12

Analyze the contingency matrix.

The contingency table for Euclidean UMAP is extremely non-diagonal and stochastic. We cannot explain this properly because of no definite trend in it. We will discuss the contingency matrix for UMAP with cosine distance metric for r=5 in this question.

Some of the ground truth clusters are getting clustered into another cluster after K-means. Some of the problematic clusters from the contingency matrix which the other clusters are getting lumped into are - 0(alt.atheism),5(comp.windows.x),12(sci.electronics),18(talk.politics.misc). On further analysis we infer that this could be due to the use of certain common words in these classes. Also, during pre processing we removed headers and footers, which could have removed some important information from the feature matrix. In addition to that, K-means is not the most ideal clustering algorithm and it has several shortcomings itself. K-means cannot handle clusters of different sizes and densities. K-means also cannot handle non-globular and non-convex cluster shapes. It is extremely sensitive to noise and outliers, and it needs a clear distinction between its boundaries.

Due to these reasons we see a percentage of 44.8% which is our highest with 20 categories data so far, as the cosine distance works extremely well with TF-IDF data, but the performance can still improve.

13 Question 13

Compare and contrast the results from the previous sections, and discuss which approach is best for the K-Means clustering task on the 20-class text data.

The results being compared in this section are between Sparse representation, PCA, NMF and UMAP with k-means clustering where n_clusters is set to 20.

The combinations tried by us are -

- SVD - With hyperparameters in the range of r = 1,2,3,5,10,20,50,100 and 300.
- NMF - With hyperparameters in the range of r = 1,2,3,5,10,20,50,100 and 300.
- UMAP - With hyperparameters in the range of r = 5, 20, 200 with euclidean and cosine distance metric.
- Sparse representation - TF-IDF matrix is used for k-means clustering.

They best performing hyper parameters from the above approaches have been summarised in the table below -

Table 4: Summary of methods with K-Means cluster size = 20

	SVD(r=100)	NMF(r=10)	UMAP (Euclidean,r=200)	UMAP (Cosine ,r=5)	Sparse
Homogenity	0.330	0.288	0.008	0.571	0.328
Completeness	0.399	0.327	0.008	0.596	0.375
V-Measure	0.362	0.306	0.008	0.583	0.350
ARI	0.111	0.099	0.001	0.448	0.116
AMI	0.359	0.304	0.005	0.582	0.347

From the table above our best performing method is UMAP with Cosine distance metric , and r = 5 applied on k-means with n_cluster = 20 with ARI close to 44.8%. The worst performing method is UMAP with Euclidean distance metric and r=200. SVD, NMF, and Sparse Representation have similar ARI scores close to 11%. From this we can conclude on our observations in the previous questions.

NMF allows only positive entries in the reduced rank feature matrix, whereas SVD has no restriction like this, because of which the results of SVD are more predictable compared to NMF. SVD is able to better represent higher-dimensional feature matrix providing a lesser information loss when compared to NMF. SVD is also more deterministic than NMF where there is a geometric basis ordered by relevance. SVD results are unique whereas, NMF is a stochastic algorithm with non-unique convergence results. That is why the results of SVD are better comparable to Sparse results, and NMF results are a bit lesser.

Cosine similarity is not affected by the magnitude of the vectors, meaning that the length of the documents does not affect the distance metric, but rather it associates clusters based on angle between sample points. This helps correct the fact that higher frequency of words in a document does not necessarily mean it belongs to a certain class given those words, it could also mean the document is very long. In addition, in high dimensions, the rapid increase in volume causes the data to become sparse in each dimension, causing the Euclidean distances to converge to a constant value between all sample points. Since all feature points become equidistant, UMAP euclidean cannot find distinguishable clusters within the data. Thus, for textual clustering, cosine similarity is a much metric when compared to L2 distances.

14 Question 14

Agglomerative clustering is also known as bottom-up approach or hierarchical agglomerative clustering (HAC). The algorithm does not require us to define number of classes. Each datapoint is considered as a singleton cluster and then successively agglomerates pairs of clusters until all clusters have been merged into a single cluster that contains all data. The linkage criteria determines the metric used for the merge strategy. The five clustering metrics for "ward" and "single" linkage criteria are as follows:

Table 5: UMAP (Cosine) with Agglomerative cluster

Metric	Agglomerative Clustering, Ward	Agglomerative Clustering, Single
Homogenity	0.557	0.015
Completeness	0.581	0.366
V-Measure	0.568	0.029
ARI	0.417	0.000
AMI	0.567	0.024

From the above table is can be seen that ward linkage criteria performs significantly better than the single linkage. In single linkage criteria we combine clusters according to minimum distance between datapoints present in the said clusters. In Ward's criteria instead of measuring the direct distance between datapoints, the variance between two clusters is measured. Ward's method says that the distance between two clusters, A and B, is how much the sum of squares will increase when we merge them. Although single linkage is fast and efficient, it fails in case of noisy data. In case of outliers single linkage method forms long snakelike chains, which causes datapoints to be clustered in wrong groups. Conversely, ward linkage criteria forms spherical tightly bound clusters and works well even on noisy data.

15 Question 15

Clusters are dense regions in the data space, separated by regions of the lower density of points. Density-based spatial clustering of applications with noise clustering algorithm is based on clusters and noise. For each point of a cluster a specified radius should contain some minimum number of data points. The algorithm has two parameters. 'eps' which defines the neighbourhood around a darapoint. If the distance between two points is less that 'eps' then they are considered as neighbors. The second parameter 'Minpts' is the minimum points within eps radius. DBSCAN groups together points based on distance metric. DBSCAN marks points in low-density regions as outliers or noise and does not group them in any cluster.
HDBSCAN extends DBSCAN by converting it into a hierarchical clustering algorithm, and then using a technique to extract a flat clustering based in the stability of clusters. The algorithm allows varying density clusters. The algorithm results in smalled tree with fewer clusters with clusters losing lesser number of points.
The two hyperparameters are epsilon and minimum cluster size.Large values of minimum samples per cluster are helpful for rejecting noise in the data. If epsilon is too small, most of the samples will be treated as outliers, while a large value erroneously merges different clusters together. The chosen values for these hyperparameters are as follows:

- epsilon = 0.5, 5
- min samples = 5, 15, 30, 60, 100, 200, 500, 1000, 3000

The best values for eps and min samples was found to be 0.5 and 100 respectively for DBSCAN. The best value for epsilon and min samples was 0.5 and 100 respectively for HDBSCAN, Using these best values the following performance metrics was found.

Table 6: DBSCAN and HDBSCAN with UMAP

Metric	DBSCAN	HDBSCAN
Homogenity	0.4812	0.4136
Completeness	0.5710	0.6257
V-Measure	0.5223	0.4980
ARI	0.2728	0.2257
AMI	0.5207	0.4971

Both DBSCAN and HDBSCAN gave similar results as can be seen from the table above. This was expected as both methods are essentially same, the only difference is ability to handle varying cluster density. If absent then there is essentially no difference in the performance of the two methods.

16 Question 16

Plot the contingency matrix for the best clustering model from Question 15. How many clusters are given by the model? What does "-1" mean for the clustering labels? Interpret the contingency matrix considering the answer to these questions.

In this question, we are asked to interpret the contingency matrices for the DBSCAN and HDBSCAN. The figures below show the contingency matrix with the best parameters obtained in the above question - DBSCAN (epsilon = 0.5, minimum sample= 100), HDBSCAN(minimum cluster size = 100, epsilon = 0.5, minimum sample = 60).

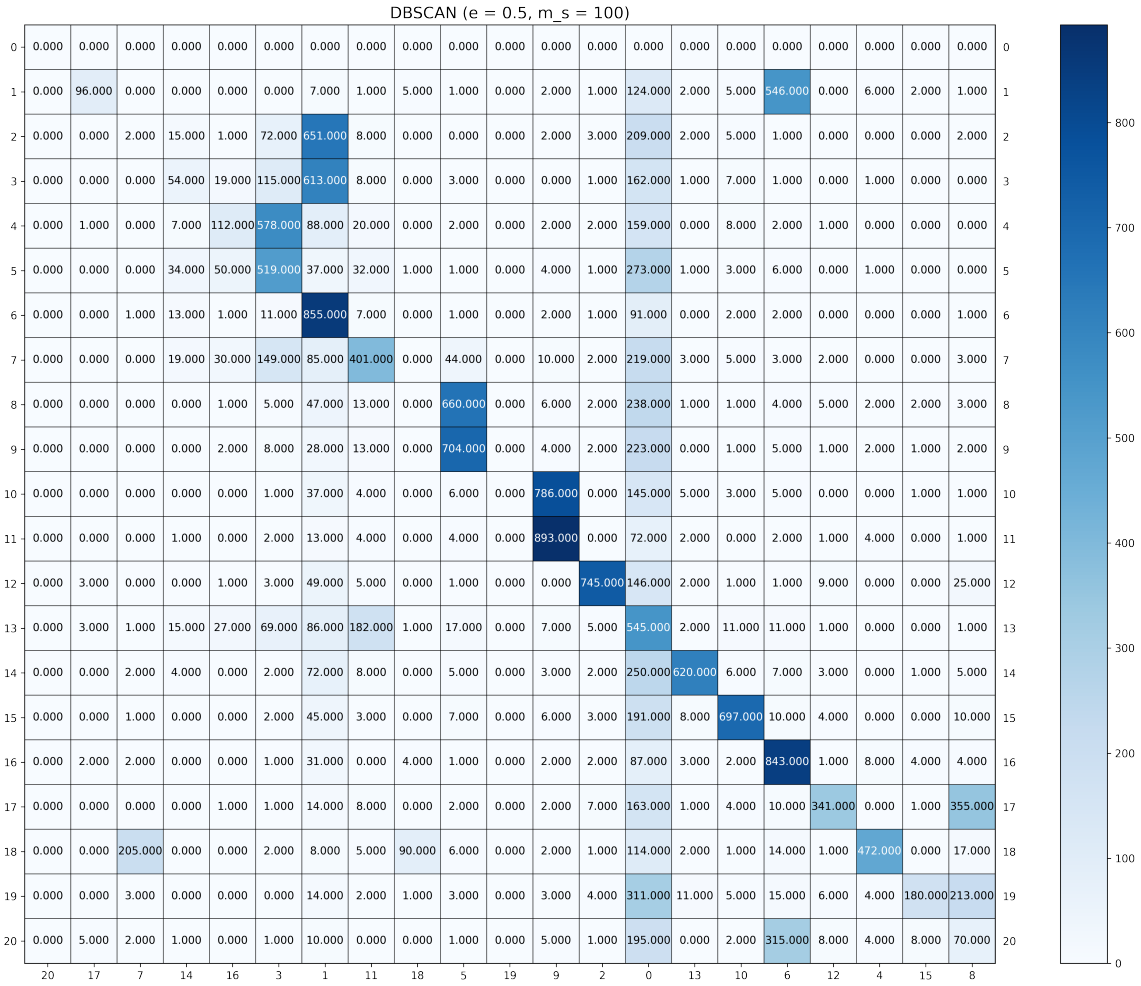

Figure 16:
Contingency matrix for DBSCAN (epsilon = 0.5, minimum sample= 100) , UMAP(Cosine r= 5)

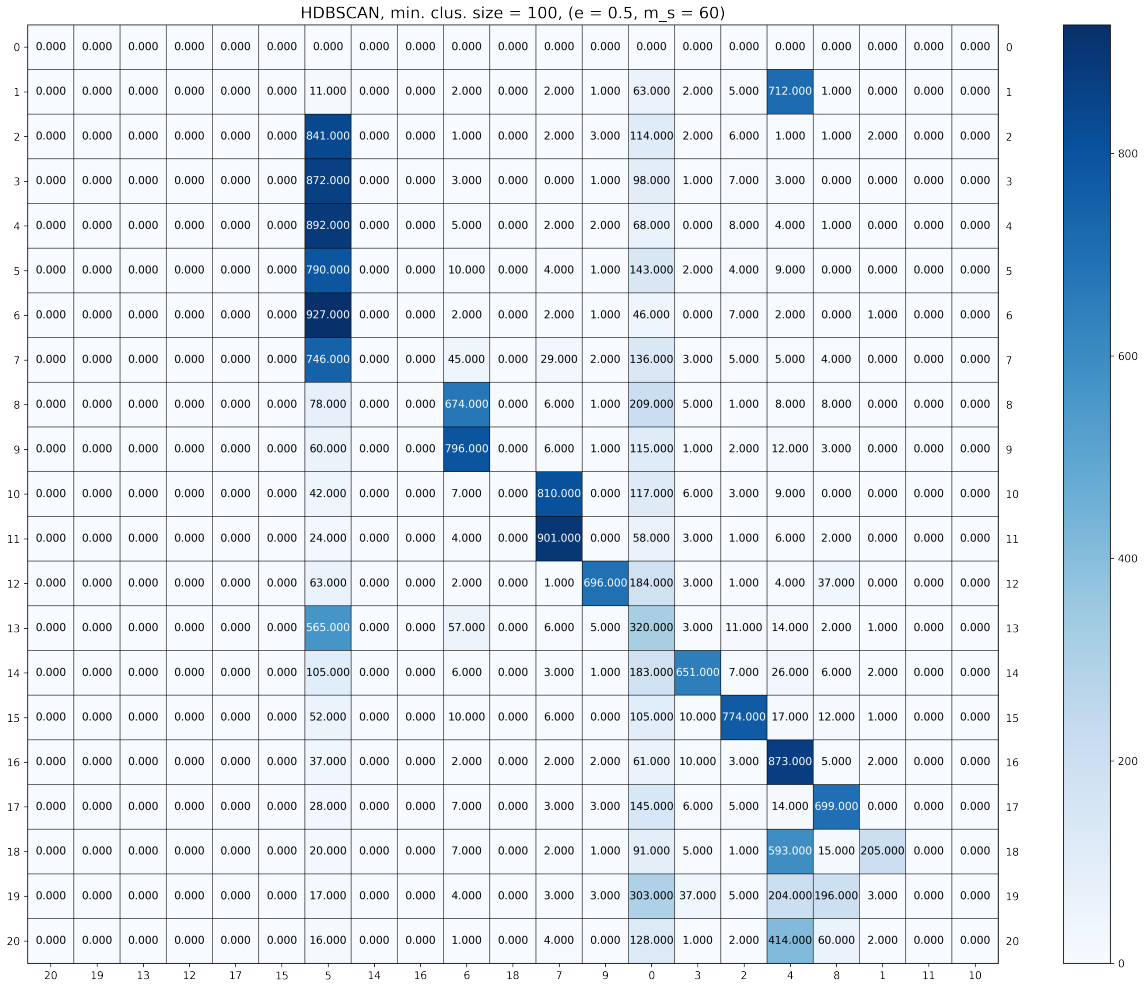

Figure 17:
Contingency matrix for HDBSCAN(minimum cluster size = 100, epsilon = 0.5, minimum sample = 60) , UMAP(Cosine r=5)

From the contingency matrices, we can see that DBSCAN produced a total of 9 major clusters (12 total), while HDBSCAN has produced a total of 8 major clusters (10 total). A lot of the missing ground truth clusters have been lumped into other clusters. The "-1" in clustering labels refers to outliers or noisy samples that have not been classified into any cluster by the algorithms. The lumping of clusters is due to excessive smoothing caused by sensitivity to the hyperparameters without altering the minimum cluster size, which has caused loss of clusters of low density. In addition, both algorithms fail to identify clusters if they are varying wildly or are sparse in high dimensions, which is indeed the case for textual data. A good option might be to experiment with the minimum cluster size hyperparameter to see if altering it leads to formation of more clusters for low-density classes, especially for HDBSCAN.

17 Question 17

Based on your experiments, which dimensionality reduction technique and clustering methods worked best together for 20-class text data and why?

In this question different combinations for dimensionality reduction and clustering have been employed, and their Adjusted Rand Index is compared. The combinations employed by us are -

- SVD - Hyperparameter r = [5,20,200]
- NMF - Hyperparameter r = [5,20,200]
- UMAP Cosine metric- n_components = [5,20,200]
- K means - Hyperparameter k = [10,20,50]
- Agglomerative Clustering - n_clusters = [20]
- DBSCAN - Hyperparameter eps = [0.5,5], minimum_samples = [5,15,100,200]
- HDBSCAN - Hyperparameter min_cluster_size = [100,200], eps=[0.5,15,20], min_samples=[5,30]
- No dimensionality reduction

The best performing results from this parameter grid search is summarised in the table below -

Table 7: Summary of Grid Search on 20-class text data

Rank	Dimensionality Reduction	Clustering	ARI
1	UMAP(r=20)	K-means(r=20)	0.461
2	UMAP(r=5)	K-means(r=20)	0.441
3	UMAP(r=5)	Agglomerative Clustering (r=20)	0.438
4	UMAP(r=20)	Agglomerative Clustering (r=20)	0.432
5	UMAP(r=200)	K-means(r=20)	0.427

With the combinations employed by us, UMAP Cosine(r=20) for dimensionality reduction with K_means (k=20) for clustering gave the highest accuracy of 46.1%.

The reason for UMAP having a good performance with k-means has already been discussed before - Cosine similarity is not affected by the magnitude of the vectors, meaning that the length of the documents does not affect the distance metric, but rather it associates clusters based on angle between sample points. This helps correct the fact that higher frequency of words in a document does not necessarily mean it belongs to a certain class given those words, it could also mean the document is very long. Thus, for textual clustering, cosine similarity is a much metric when compared to L2 distances, specially in the higher dimensions when data starts becoming more sparse.

The advantages of using K-means includes that the algorithm eventually converges given enough time. It scales linearly in time for large datasets and adapts to new examples on-the-fly (unsupervised). Also, it has a fast results delivery. In this question the number of categories is also 20, so intuitively k=20 giving the best performance makes perfect sense.

The ward linkage criteria in agglomerative clustering is similar to k-means clustering and it encourages formation of spherically tightly bound clusters and works very well in presence of noise.

18 Question 18

Creative ways to further enhance the clustering performance,report your method and the results you obtain.

We tried the following combinations -

- Using word normalisation with CountVectorizer(). More specifically lemmatization is performed using the pos_tag. This algorithm takes into consideration the morphological analysis of words. It depends on part-of-speech information, and different rules are applied based on the semantic information of the word - whether it is a noun, verb or adjective. Lemmatization reduces variability of different forms of the same stem word, and hence decreases the size.

  This was used with UMAP(Cosine, r=20 for dimensionality reduction and K-means(k=20) for clustering. The ARI obtained from this method is - 0.16.

- We tried including the headers and footers in the data, and use the best model of UMAP(Cosine, r=20) with k-means(k=20) combination on this. The ARI obtained from this method is - 0.41.

19 Question 19

The VGG network is trained on ImageNet consisting of 1000 classes, the class 'daisy' matches the class given to us in the tensorflow flower dataset. VGG network is trained on dataset which has target classes different than our custom dataset.
In deep learning, more training data leads to better classifying models. Imagenet has over 1.2 million labeled images in 1000 different classes. Although these classes do not include the target classes in tf-flower dataset, it still has discriminative power for the following reasons:

- In tranfer learning, the model learns to extract common features as it has been trained on more than 1.2 million images from 1000 different classes which are sufficient to extract generic features from custom dataset and classify then into target datasets.
- Considering the example of testing, when we test on a specific image the model does not need to look back at the images it trained on before, just looking at the features extracted it can classify the image. Similar to the example in transfer learning we just change the output layer so that we get cusotm target labels. The pretrained network learns features in the new images and classifies them accordingly.
- As we have some flower classes in the imagenet dataset(including common class Daisy) we can say that vgg16 network is able to extract features from flower images and differentiate them. Hence, we can say that features derived from imagenet will have discriminative power for our custom dataset.

20 Question 20

The images from flower dataset are loaded. The images from the dataset have different dimensions. Hence resizing in VGG network helps. All the images from the dataset are first resized to 224 x 224 as the VGG network has a fixed input of 224 x 224. Once the images are resized we also center crop the images in the same size.

The data is then stored as a tensor and normalized with the specified mean and standard deviation. In preprocessing step, mean value of rgb is subtracted from each pixel of the training images.

In VGG network, the image is then passed through a stack of conv layers, where filters with very small receptive fields such as 3x3 are used. 3 x 3 is the smallest size of fliters which can capture the notion of left, right, center etc. Spatial pooling is performed by five max-pooling layers which follow some conv layers. This is followed by two fully connected layers which have 4096 output channels.The final fully connected layer gives 1000 channels as output, one for each class of imagenet. We do not use this last layer. We just include the first two fully connected layers and get 4096 channels as output.

We are importing VGG-16 pretrained of Imagenet dataset. We will sequentially pass all the images through thefeature layers in VGG network. The feature maps will be then passed through a pooling layer. The images will then be passed through a flattening layer which will covert the images to a 1D vector. Then, fully connected layer which will give 4096 features points.

21 Question 21

Each image in the custom dataset has a different number of pixels. Choosing a few random images we can see the image size to be as follows and total number of pixels in the table below.

Table 8: Number of pixels in randomly chosen samples

Target Label	Image Size	Number of pixels
Daisy	320 x 232	74240
Dandelion	320 x 240	76800
Rose	320 x 240	76800
Sunflower	180 x 240	43200
Tulip	500 x 333	166500

There are many images in the dataset with the size of 240*240. Although the size for each image varies. Due to this, resizing in VGG is very effective. The VGG network outputs a feature vector of size 4096 for each image.

22 Question 22

The extracted features are dense as they contain less number of zeroes. Also, as we extract features from images we convert high dimensional data into lower dimensional data then it becomes dense. In case of tf-idf features as we have high dimensional data which we store in tf-idf feature vectors it actually gets filled up with zeros and it becomes sparse matrix.

23 Question 23

From the plots below, it can be observed that the clusters are visible in t-SNE plot while not very clearly visible in the pca one. T-SNE is one of the best dimensionality reduction techniques where as pca does not work as well as t-SNE for the following reasons.
PCA is an unsupervised linear dimensionality reduction and data visualization technique for very high dimensional data. High dimensional data is computationally intensive and it is very hard to gain insights from it. The main idea is to reduce highly correlated data into new vectors known as principle components. PCA tries to preserve global structure of the data hence local structure may be lost, which can be observed in the image shown. Hence PCA gets highly affected by outliers. The algorithm works by rotating vectors to preserve the variance.
t-distributed stochastic neighbourhood embedding (t-SNE) is a unsupervised non-linear dimensionality reduction and data visualization technique.The fundamental idea is that it embeds a point from higher dimension to lower dimension while preserving the neighborhood of that point, hence clusters are more visible in case of t-SNE as shown in the plot below. Unlike PCA, t-SNE preserves the local structure instead of global structure. So, outliers do not affect the method. The algorithm works by minimizing distance between the points in a gaussian.
As explained by the reasons above and can be visibly seen by the plots below while t-SNE is one of the best dimensionality reduction technique, pca does not work as well as t-SNE.

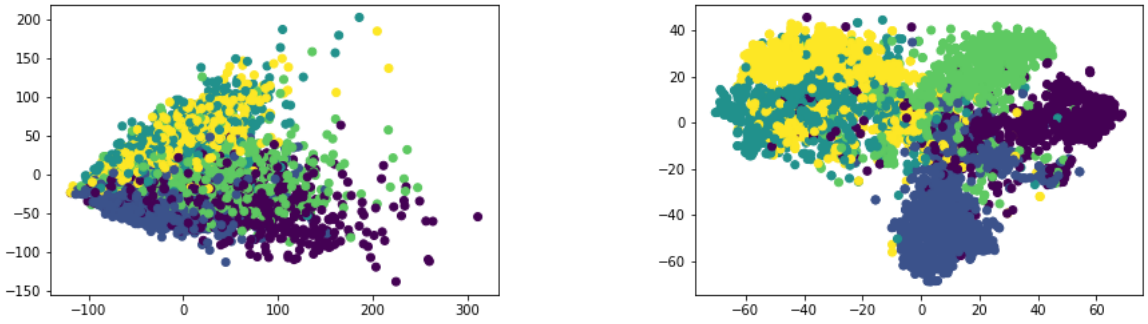

Figure 18: (Left) PCA (Right) TSNE

24 Question 24

Table 9: Performance Metric for Linear SVM Classifier with GLOVE embeddings

Clustering method	None	SVD	UMAP	Autoencoder
K-Means	0.1917405082	0.1900539537	0.4353405321	0.2749528971
Agglomerative Clustering	0.1885527825	0.163897373	0.3621791785	0.2454631794
hdbscan($\min_c lustersize = 5, minsamples = 1$)	0.01498303459	0.02354639934	0.01255579639	0.01956262812
hdbscan($\min_c lustersize = 100, minsamples = 30$)	0	0	0.09116303825	0
hdbscan($\min_c lustersize = 100, minsamples = 5$)	0	0	0.3469239034	0
hdbscan($\min_c lustersize = 200, minsamples = 30$)	0	0	0.2409005007	0
hdbscans($\min_c lustersize = 10, minsamples = 1$)	0.015014212771105666	0.01641908904146645	0.20985292346172957	0.021902594900760027

The Rand Index computes a similarity measure between two clustering methods by considering all pairs of samples and counting pairs that are assigned in the same or different clusters in the predicted and true clusters. For adjusted rand score, 0 means the samples are assigned to clusters randomly. When rand score is 1 means all samples are assigned correctly. It can be inferred that more the value of rand score the better the results. So, in our example, Kmeans with umap used as dimensionality reduction performs best with 0.43 adjusted rand score.

The reason for UMAP having a good performance with k-means has already been discussed in sections above - Cosine similarity is not affected by the magnitude of the vectors, but rather it associates clusters based on angle between sample points. Thus, cosine similarity is a much metric when compared to L2 distances.

The advantages of using K-means includes that the algorithm eventually converges given enough time. It scales linearly in time for large datasets and adapts to new examples on-the-fly (unsupervised). Also, it has a fast results delivery.

As a result K-means clustering with UMAPs gives the best result in all the clustering algorithms.

25 Question 25

Table 10: Performance Metric for Linear SVM Classifier with GLOVE embeddings

Dimension Reduction method	Test Accuracy
None	90.87
umap	86.64
Autoencoder	89.37
t-SNE	77.92
PCA	56.40

- As can be seen from the table the performance of MLP network gets affected by reduced features significantly. Classification results are better when trained on all features. The performance worsens on reduced features ranging from 1 to 35 percent depending on the reduction method used.
- To compare performance of clustering methods with classification we can consider the metric we have already calculated, as adjusted Rand Index is similar to accuracy. Adjusted Rand Index computes similarity between the clustering labels and ground truth labels. This method counts all pairs of points that both fall either in the same cluster and the same class or in different clusters and different classes. Even though adjusted rand index and accuracy are comparable they are still different terms.
- In case of clustering dimension reduction methods improved the performance. Whereas, in case of mlp network dimension reduction worsened the performance.
- Classification performs considerably well on the custom dataset giving 90.87 accuracy for no dimension reduction. For dimension reduction the performance decreases depending on the method used. Whereas clustering the best adjusted rand score(comparable to accuracy) is found to be 0.43 which is considerably lower value compared to classification accuracy. Also, clustering performs better with dimensionality reduction.
- As we have the labels of the dataset supervised learning works better compared to unsupervised learning.