ECE219 Large-Scale Data Mining: Models and Algorithms **(Due: Sun, 02/11/24)**

Project 2: Data Representations and Clustering

Instructor: Vwani Roy Chowdhury Aryaman Rajesh Gokarn 506303588

I. 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.
- Used TfidfVectorizer which acts as both a CountVectorizer() and then converts the output of that into a TF-IDF matrix. The CountVectorizer() converts 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. The TF-IDF matrix provides more useful and distinguishing features over bag-of-words for the classifier we want to use.

II. 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.

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

$$r_{nk} = \begin{cases} 1, & \text{if } \mathbf{x}_n \text{ is assigned to cluster } k \\ 0, & \text{otherwise} \end{cases}, \quad n = 1, \ldots, N \quad k = 1, \ldots, K$$

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.

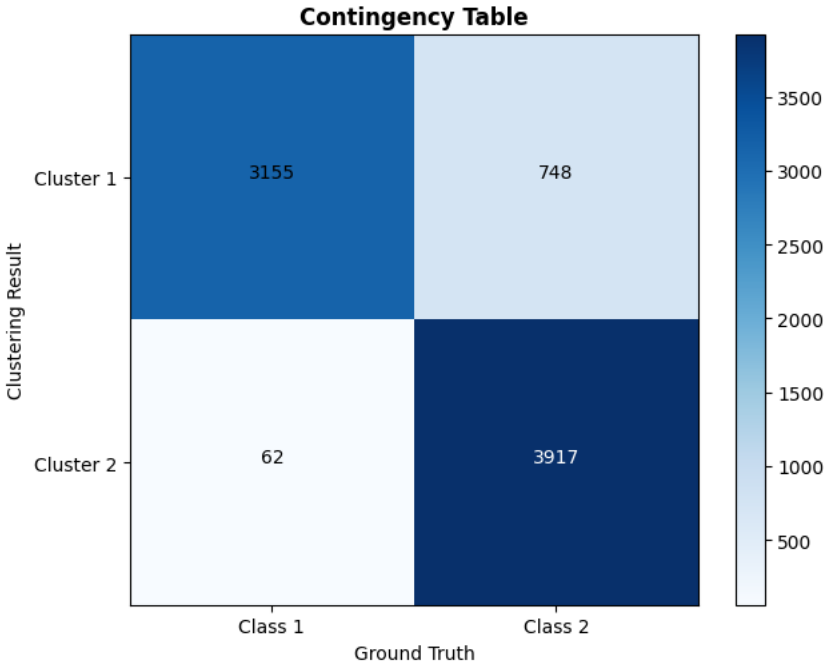


Figure 1:

Contingency matrix for K=2 clusters on TF-IDF matrix

It's important to note that the **contingency matrix may not be squar**e if the number of classes in the true labels is not equal to the number of clusters obtained from the clustering algorithm. The matrix will have dimensions equal to the number of unique classes in the true labels (rows) and the number of clusters (columns). If the number of classes and clusters is the same, the contingency matrix will be square. However, in many real-world scenarios, the number of clusters may not necessarily match the number of true classes.

III. 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 Score: 0.568013
- Completeness Score: 0.582230
- V-Measure Score: 0.575034
- Adjusted Rand Index: 0.631134
- Adjusted Mutual Information: 0.574995

From these results we observe that the homogeneity is approximately 57%, which indicates that 43% of the samples in a particular cluster are not from the assigned ground truth label. The completeness is also approximately 58%, which indicates that 42% 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 57% respectively).

IV. 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

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

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 componentsset 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%.

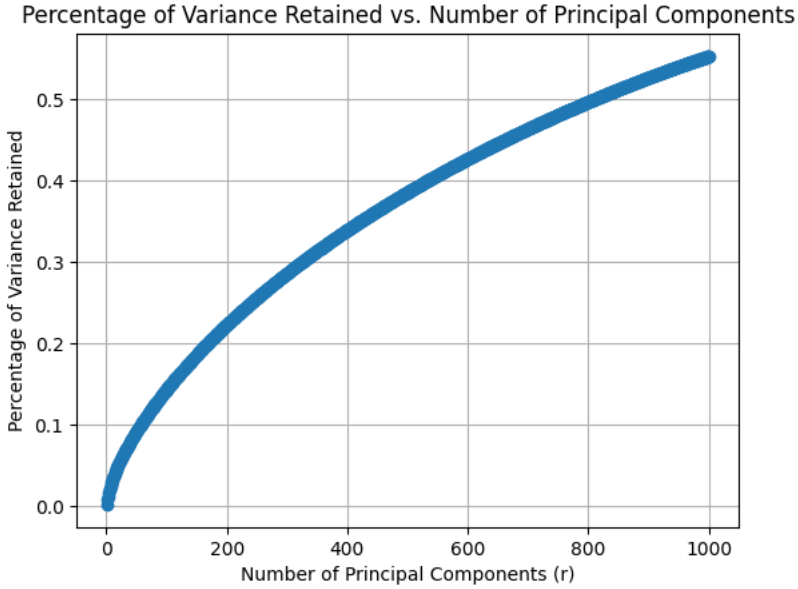


Figure 2:
Plot of the percentage of variance that the top r principle components retain v.s. r, for r = 1 to 1000

V. 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 $\mathbf{X}$ as possible, with W being the low-rank feature matrix for $\mathbf{X}$, 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 forthe best r value based on these plots.

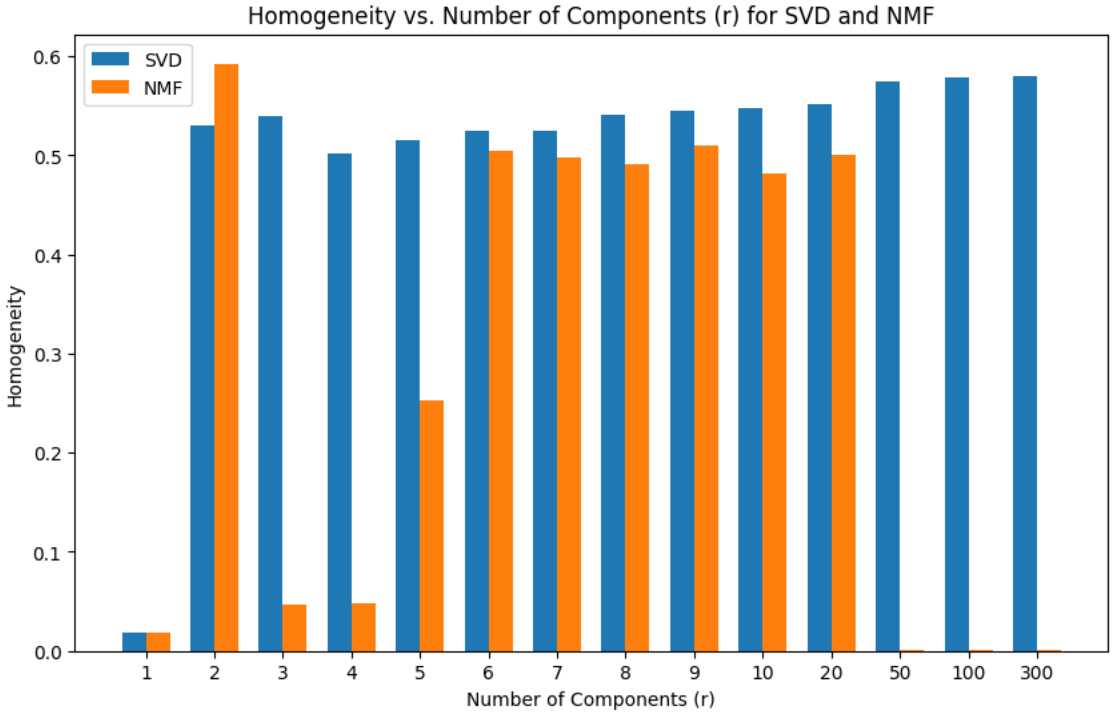


Figure 3: Plot of Homogeneity Score v/s r

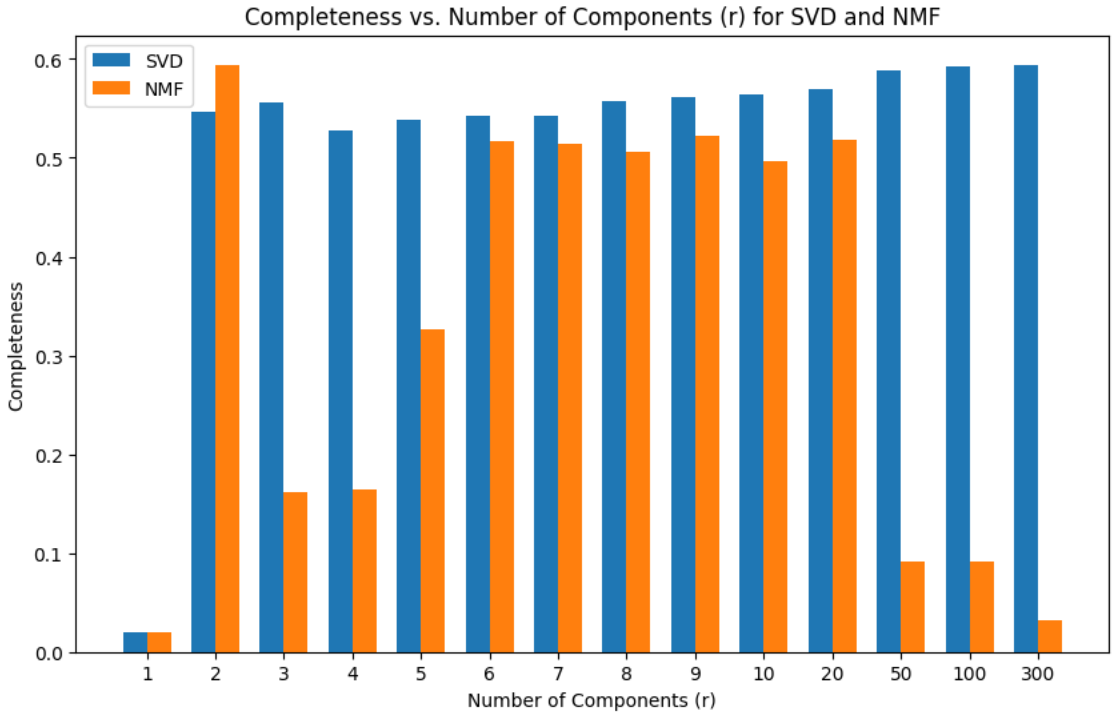


Figure 4: Plot of Completeness Score v/s r

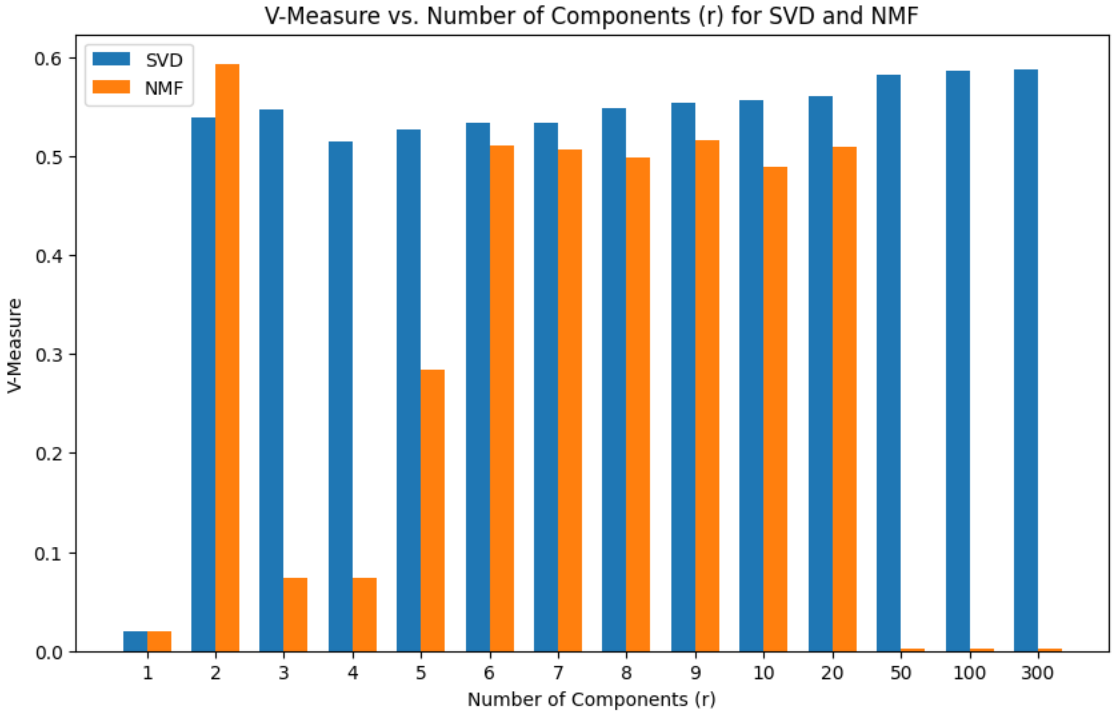


Figure 5: Plot of V-Measure v/s r

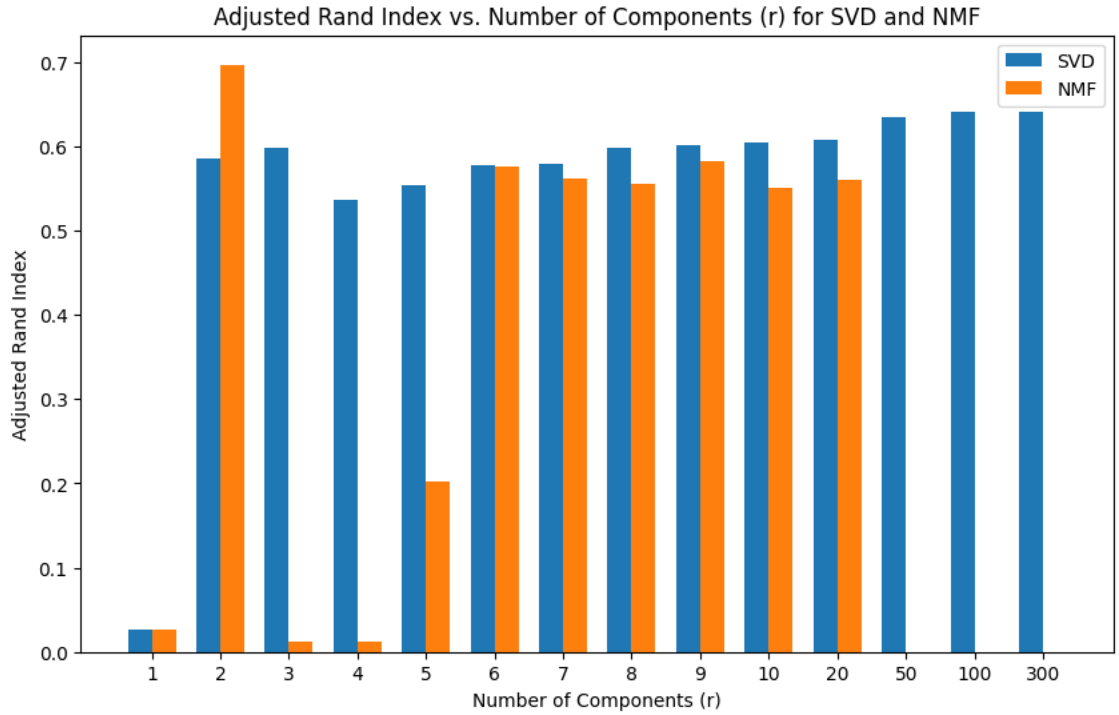


Figure 6: Plot of Adjusted Rand Index Score v/s r

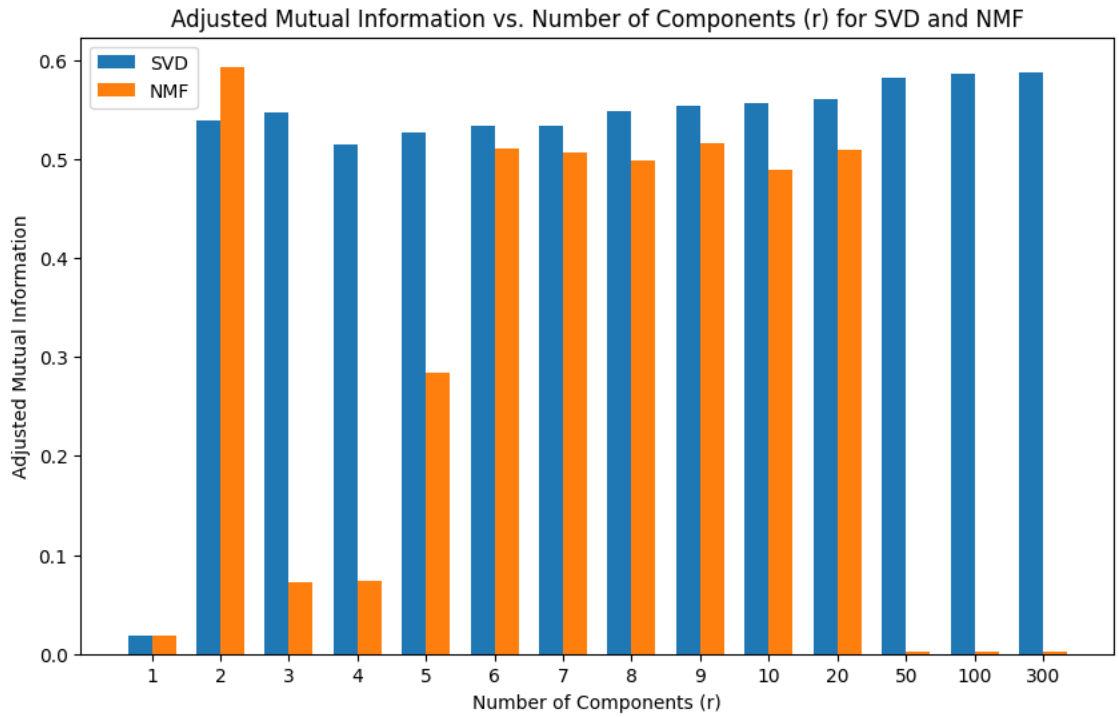


Figure 7: Plot of Adjusted Mutual Information v/s r

Therefore we can see:-

- A good choice of r for SVD: 300
- A good choice of r for NMF: 2

VI. 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 com- pared 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.

VII. 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 Score: 0.568013
- Completeness Score: 0.582230
- V-Measure Score: 0.575034
- Adjusted Rand Index: 0.631134
- Adjusted Mutual Information: 0.574995

Measures computed from highest parameter of SVD after feature reduction (n components=300):-

- Homogeneity: 0.578467283972509
- Completeness: 0.5930703687890024
- V-Measure: 0.5856778134774765
- Adjusted Rand Index: 0.6388183290014525
- Adjusted Mutual Information: 0.5856394089690624

Measures computed from best parameters of NMF after feature reduction (n components=2):-

- Homogeneity: 0.5924736270587376
- Completeness: 0.5939066206393134
- V-Measure: 0.5931892584137631
- Adjusted Rand Index: 0.6968751544445039
- Adjusted Mutual Information: 0.5931519697549141

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 and NMF peforms almost similar to but slightly better than the sparse data when n components chosen is 300 and 2 respectively. All the 5 metrics are very similar.

VIII. Question 8

Visualize the clustering results for:
• SVD with your optimal choice of r for K-Means clustering;
• NMF with your choice of r for K-Means clustering.

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 300 for SVD and 2 for NMF. Figure 8,9 shows the plot for SVD with ground truth labels and k-means labels while Figure 10,11 shows the plot for NMF with ground truth labels and k-means labels.

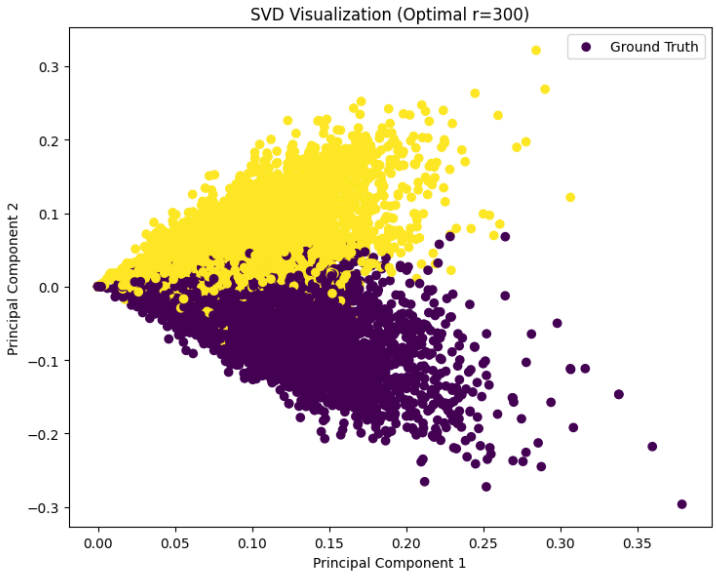


Figure 8: SVD with GT labels for r = 300

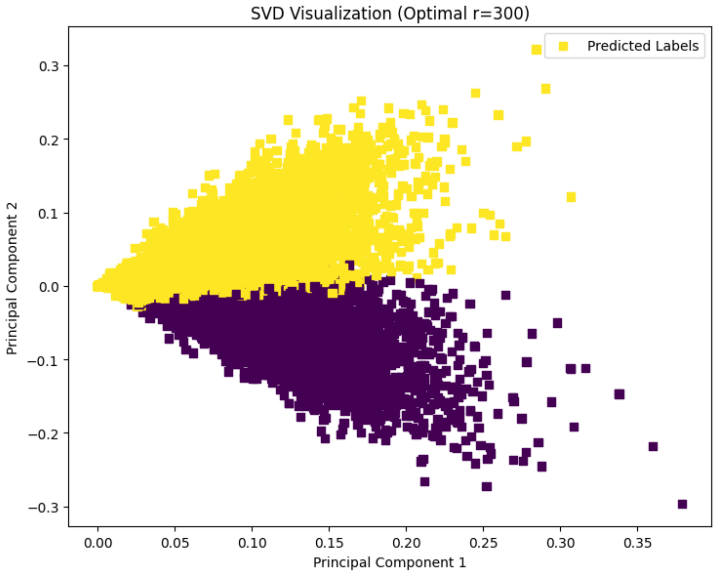


Figure 9: SVD with K-means labels for r = 300

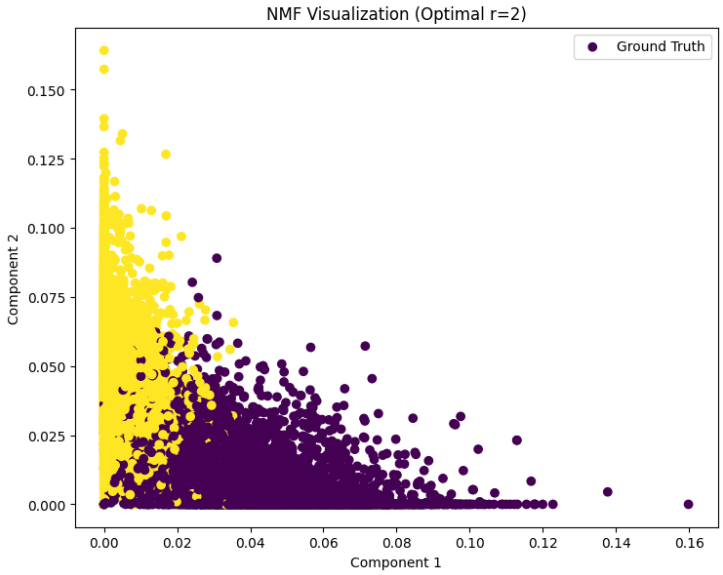


Figure 10: NMF with GT labels for r=2

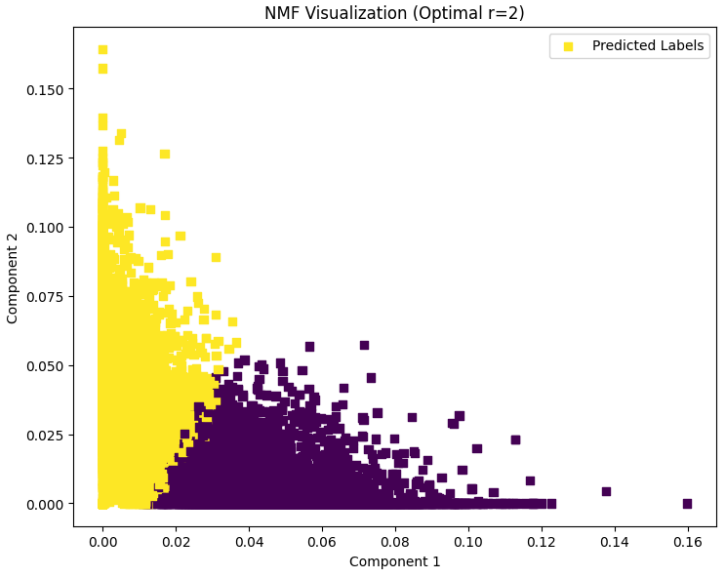


Figure 11: NMF with k-means labels for r=2

IX. 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 visualization 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.
- 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. 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.

X. 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.
- Used TfidfVectorizer() 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 TfidfVectorizer(). 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 and NMF. We used the best hyperparameter value of r for each case obtained in the previous question.
- 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.

The clustering metrics reported on that are –

- **Results for SVD-reduced:**
  - Homogeneity Score: 0.33442534127919826
  - Completeness Score: 0.41206736205624855
  - V-Measure: 0.36920861401576205
  - Adjusted Rand Index: 0.10026574460864296
  - Adjusted Mutual Information: 0.3669344051407069
- **Results for NMF-reduced:**
  - Homogeneity Score: 0.18969353513135712
  - Completeness Score: 0.20111391114511892
  - V-Measure: 0.19523685709008776
  - Adjusted Rand Index: 0.055708129841347376
  - Adjusted Mutual Information: 0.19255689533801038

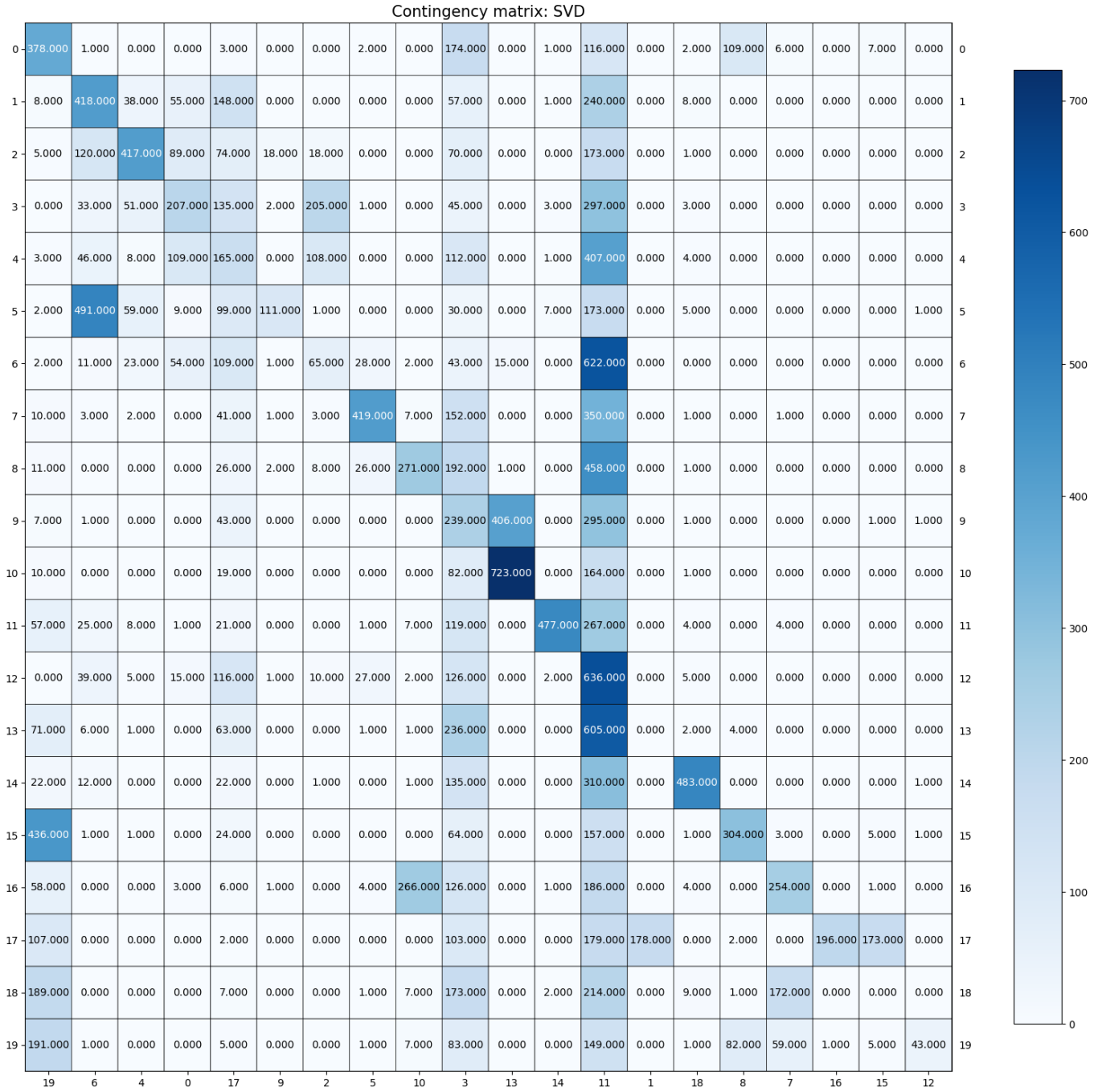


Figure 12:

Contingency matrix with SVD r= 300, and K-means cluster= 20

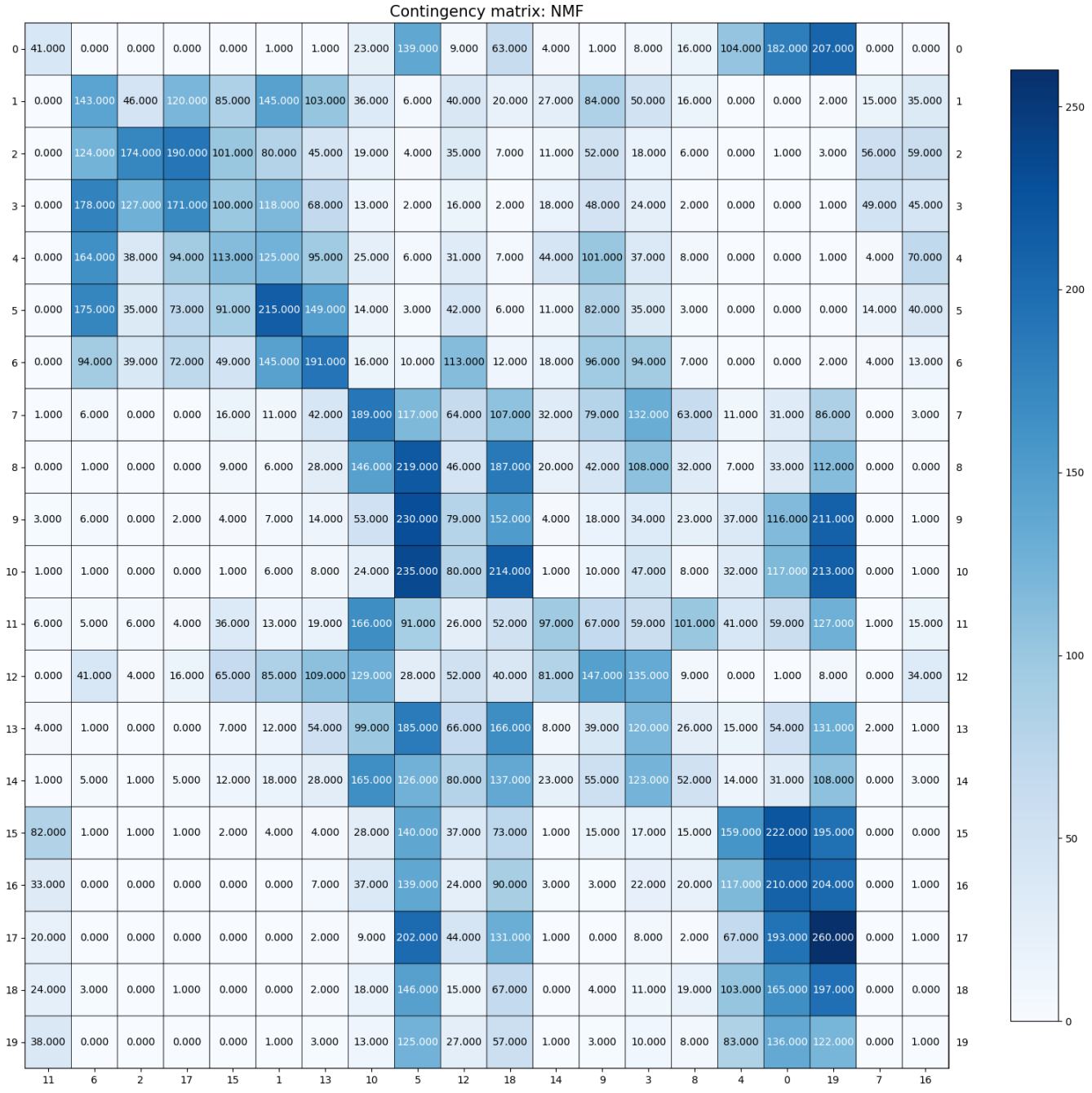


Figure 13:

Contingency matrix with NMF r= 2, and K-means cluster= 20

XI. 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 represent- ing 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 struc- ture.. 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
Homogeneity	0.009	0.009	0.008
Completeness	0.009	0.009	0.008
V-Measure	0.009	0.009	0.008
ARI	0.001	0.001	0.001
AMI	0.006	0.005	0.004

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

	r=5	r=20	r=200
Homogeneity	0.566	0.561	0.578
Completeness	0.593	0.586	0.600
V-Measure	0.597	0.573	0.589
ARI	0.453	0.428	0.463
AMI	0.578	0.572	0.587

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 200. The reported clustering metrics for these values are -

UMAP Euclidean:

- Homogeneity: 0.007537499073160126
- Completeness: 0.007836783401006145
- V-measure: 0.007684228219547366
- Adjusted Rand-Index (ARI): 0.000996877893881684
- Adjusted Mutual Information Score (AMI): 0.004423629962601411

UMAP Cosine:

- Homogeneity: 0.5780130176801029
- Completeness: 0.6004255407994405
- V-measure: 0.5890061492513815
- Adjusted Rand-Index (ARI): 0.4633960903061091
- Adjusted Mutual Information Score (AMI): 0.5876519472124563

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 46% compared to 0.1%. This is expected behavior 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.

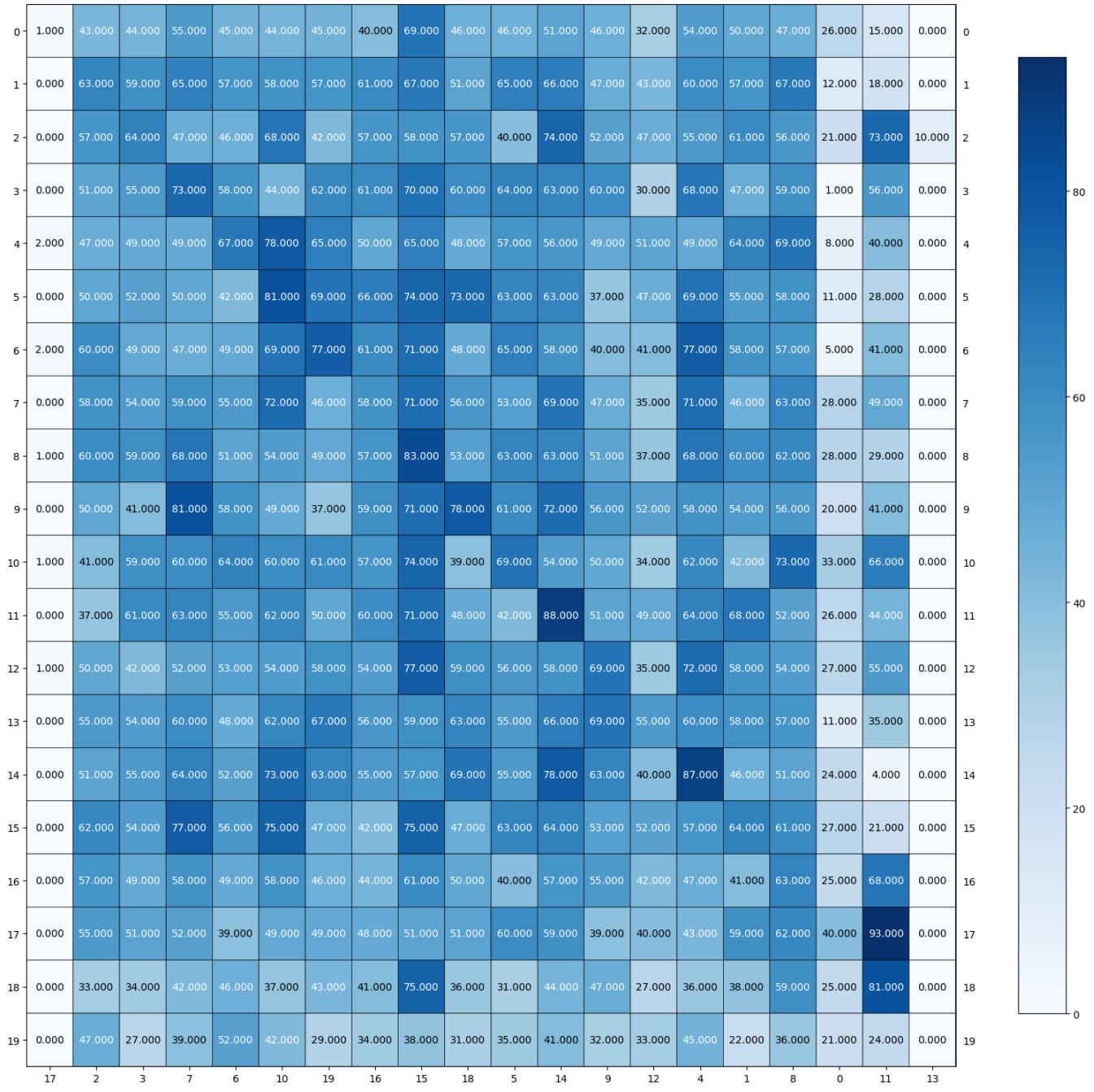


Figure 14:
Contingency matrix with UMAP (euclidean) r= 200, and K-means cluster= 20

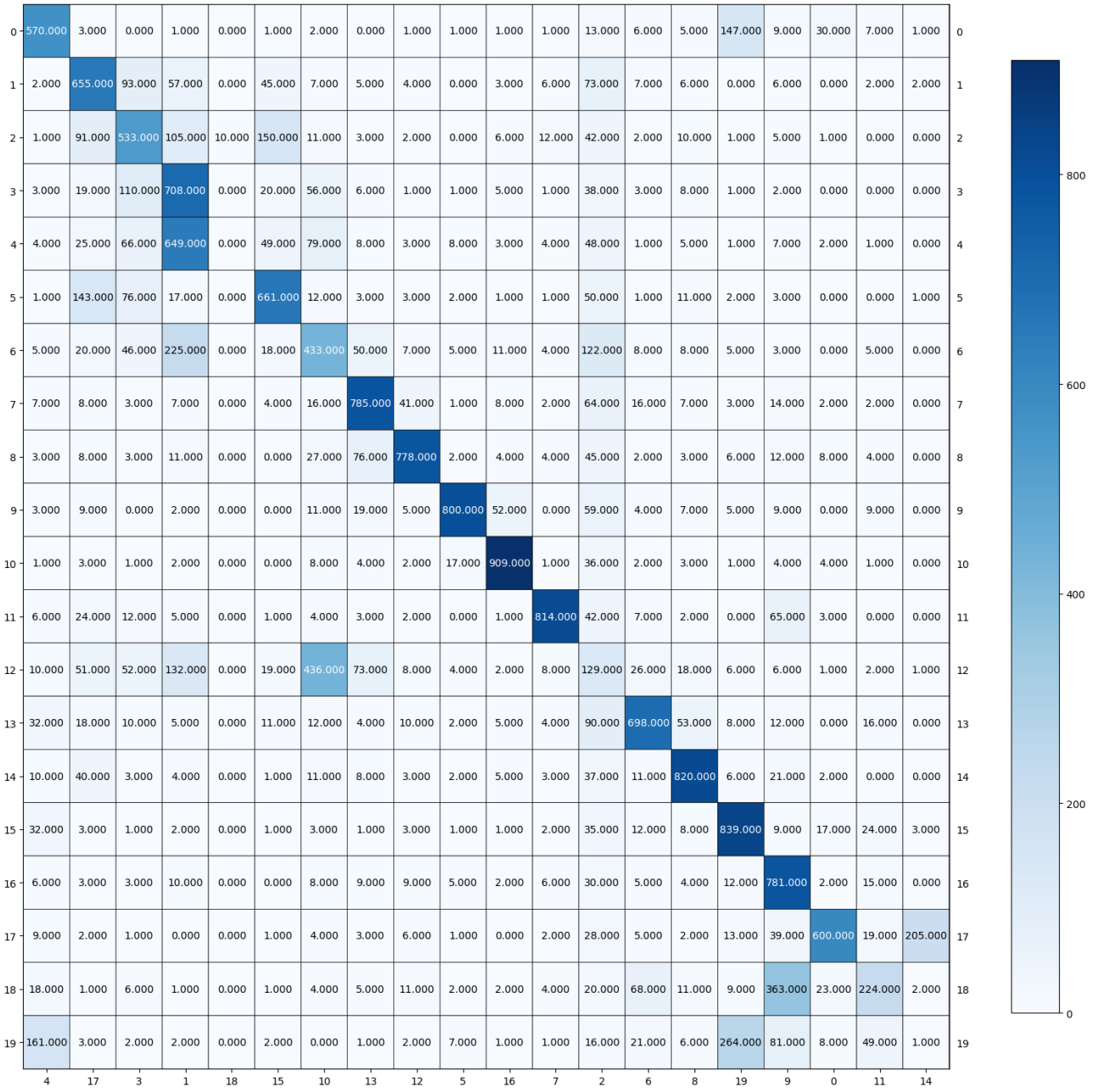


Figure 15:
Contingency matrix with UMAP (cosine) r= 200, and K-means cluster= 20

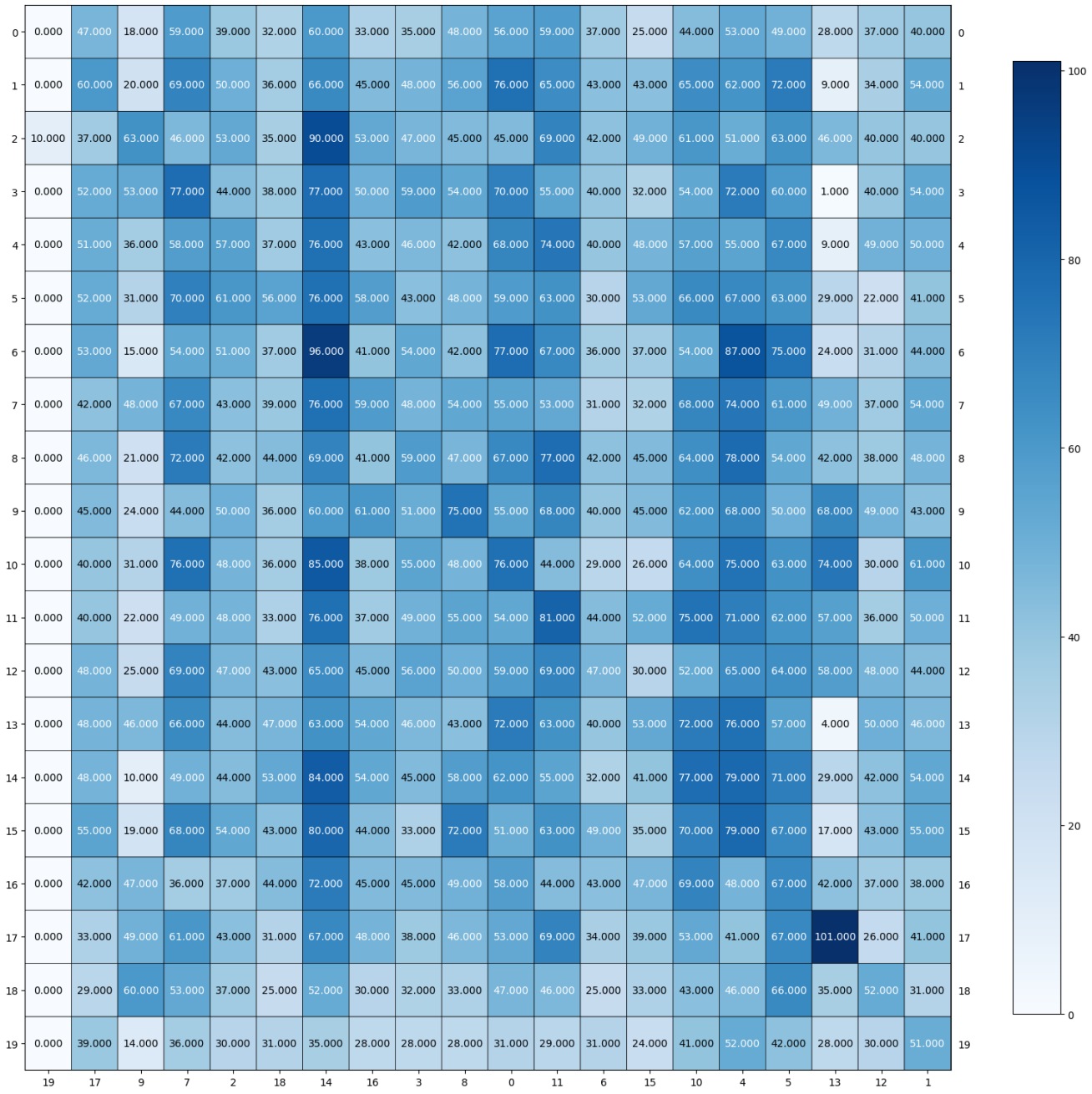


Figure 16:
Contingency matrix with UMAP (Euclidean) r= 5, and K-means cluster= 20

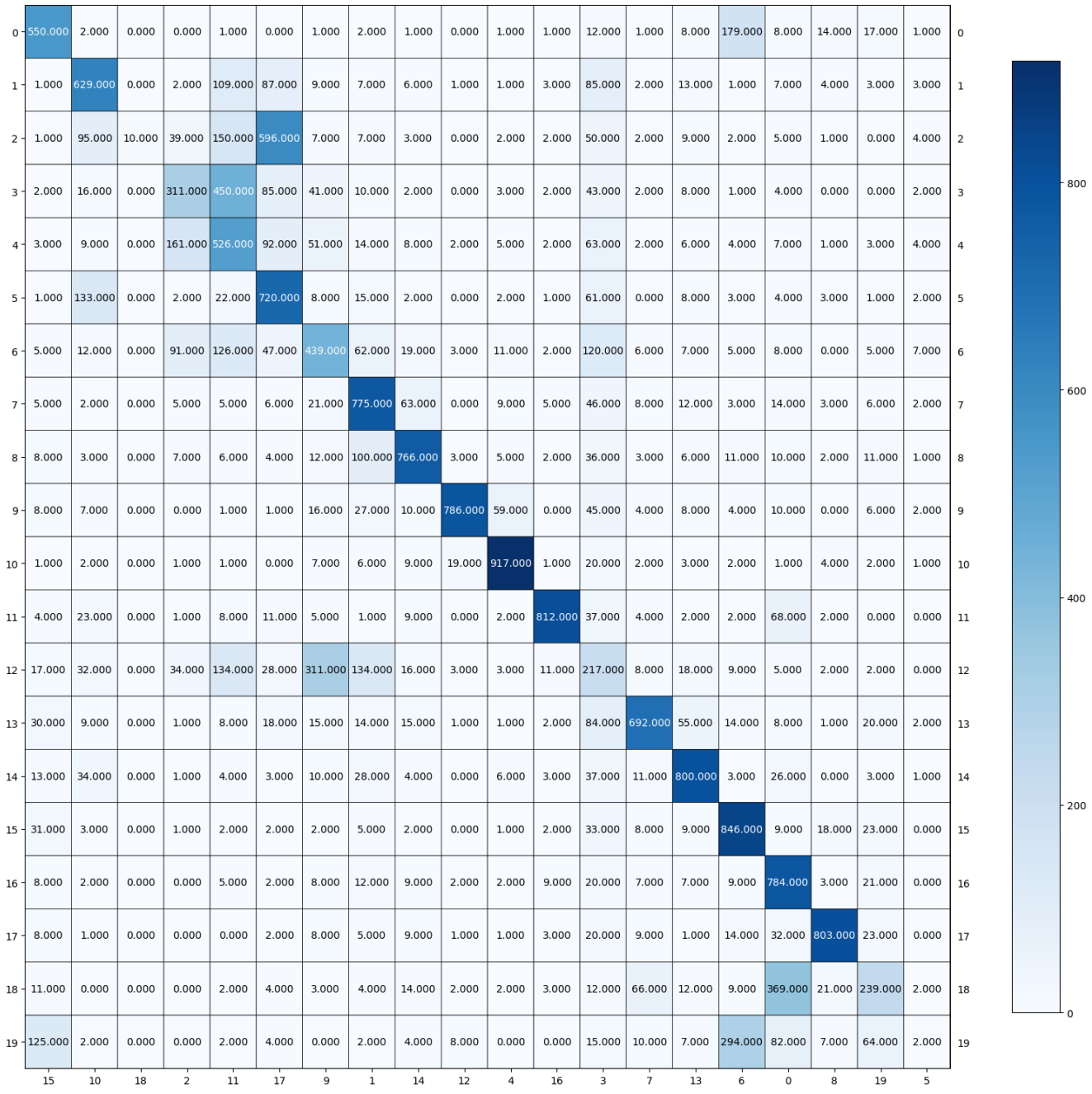


Figure 17:
Contingency matrix with UMAP (cosine) r= 5, and K-means cluster= 20

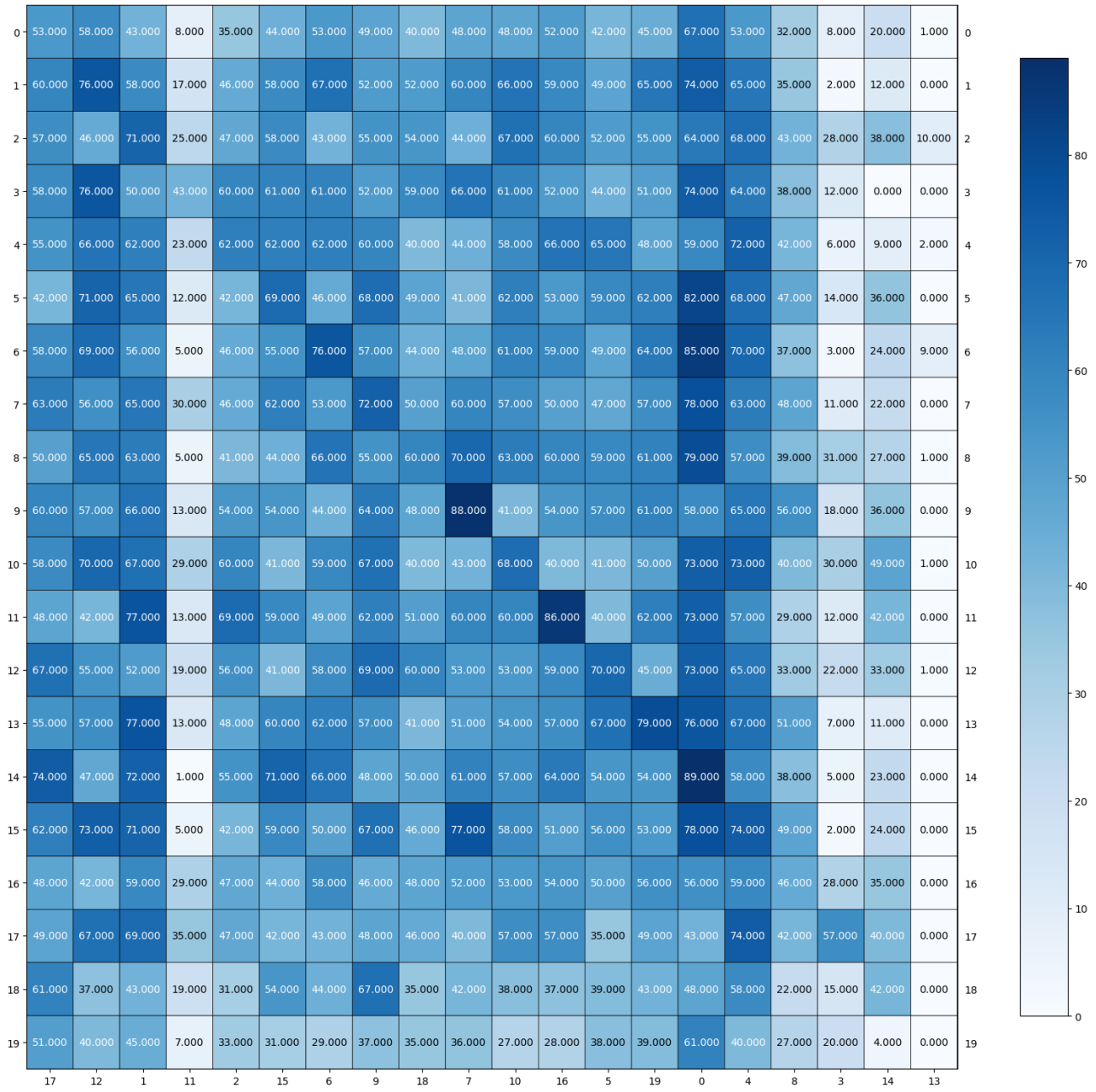


Figure 18:
Contingency matrix with UMAP (Euclidean) r= 20, and K-means cluster= 20

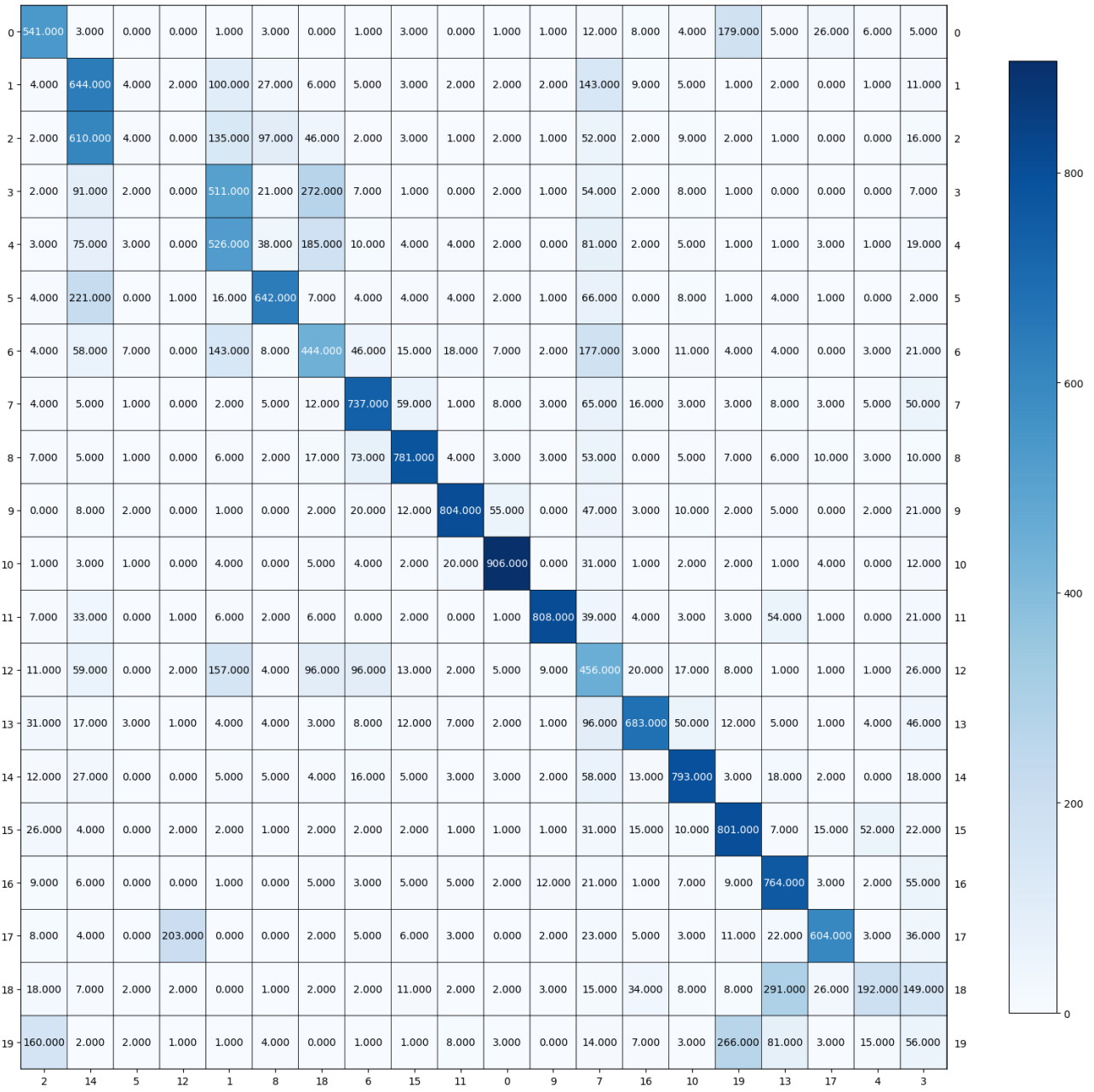


Figure 19:
Contingency matrix with UMAP (cosine) r= 20, and K-means cluster= 20

XII. Question 12

Analyze the contingency matrices. Which setting works best and why? What about for each metric choice?

Best Accuracy is given by: UMAP Settings: n_components=200, metric=cosine

Homogeneity: 0.5780130176801029
Completeness: 0.6004255407994405
V-Measure: 0.5890061492513815
Adjusted Rand Index: 0.4633960903061091
Adjusted Mutual Information: 0.5876519472124563

In the context of Euclidean UMAP, the contingency matrix exhibits a non-diagonal and dispersed pattern. On the other hand, in the case of cosine UMAP, certain ground truth clusters are consolidated into a single cluster, resulting in a diagonal pattern. Overall, it is evident that cosine, when used with various components, outperforms Euclidean as a metric. The diagonal structure in cosine indicates a more accurate mapping of the majority of clusters, contrasting with the less favorable mapping observed in Euclidean.

XIII. Question 13

Compare and contrast the clustering results across the 4 choices and suggest an approach that is best for the K-Means clustering task on the 20-class text data. Choose any choice of clustering metrics for your comparison.

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.

They best performing hyper parameters from the above approaches have been summarized in thetable below -

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

	Sparse TFIDF	SVD (r=300)	NMF (r=2)	UMAP (Cosine, r=200)
Homogeneity	0.371	0.334	0.189	0.578
Completeness	0.428	0.412	0.201	0.600
V-Measure	0.397	0.369	0.195	0.589
ARI	0.136	0.100	0.055	0.463
AMI	0.395	0.366	0.192	0.587

From the table above our best performing method is UMAP with Cosine distance metric, and r = 200 applied on k-means with n cluster = 20 with ARI close to 46%. The worst performing method isNMF and r=2. SVD, and Sparse Representation have similar ARI scores close to 10-13%.

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 con- vergence 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 be-tween 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 distinguish- able clusters within the data. Thus, for textual clustering, cosine similarity is a much metric when compared to L2 distances.

XIV. Question 14

Use UMAP to reduce the dimensionality properly and perform Agglomerative clustering with n_clusters=20. Compare the performance of “ward” and “single” linkage criteria. Report the five clustering evaluation metrics for each case.

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 determine 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
Homogeneity	0.564	0.016
Completeness	0.589	0.365
V-Measure	0.576	0.032
ARI	0.433	0.001
AMI	0.575	0.027

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.

XV. Question 15

Apply HDBSCAN on UMAP-transformed 20-category data. Use min_cluster_size=100. Vary the min cluster size among 20, 100, 200 and report your findings in terms of the five clustering evaluation metrics - you will plot the best contingency matrix in the next question. Feel free to try modifying other parameters in HDBSCAN to get better performance.

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.

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 = 0.01,0.05, 0.1, 0.15, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0,3.0,5.0,10.0, 30.0,50.0
- min samples = 5, 15, 30, 60, 100, 200, 500, 1000, 3000

The best values for eps and min samples was found to be 0.3 and 5 respectively for HDBSCAN for cluster size 20 and 100. The best value for epsilon and min samples was 0.01 and 500 respectively for HDBSCAN for cluster size 200, Using these best values the following performance metrics was found.

Table 6: HDBSCAN with UMAP

Metric	min_cluster_size=20	min_cluster_size=100	min_cluster_size=200
Homogeneity	0.494	0.445	0.376
Completeness	0.427	0.514	0.592
V-Measure	0.458	0.477	0.460
ARI	0.259	0.228	0.156
AMI	0.446	0.475	0.459

XVI. Question 16

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.

Model with the best average metrics was the model with a **minimum cluster size of 100**

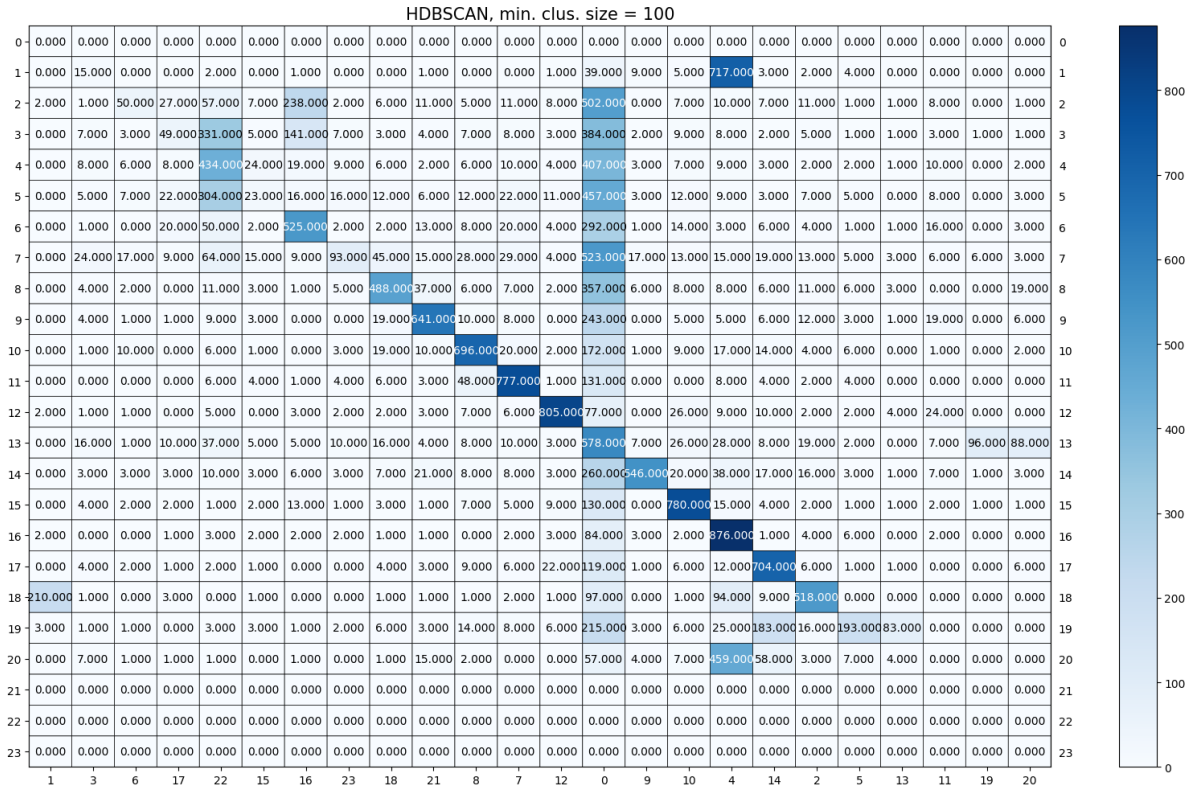


Figure 20:

Contingency matrix for HDBSCAN (minimum cluster size = 100, epsilon = 0.3, minimum sample =5), UMAP (Cosine r=200)

From the contingency matrices, we can see that 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, the algorithm fails to identify clusters if they are varying wildly or are sparse in high dimensions, whichis indeed the case for textual data.

XVII. Question 17

Based on your experiments, which dimensionality reduction technique and clustering methods worked best together for 20-class text data and why? Follow the table below. If UMAP takes too long to converge, consider running it once and saving the intermediate results in a pickle file.

Module	Alternatives	Hyperparameters
Dimensionality Reduction	None	N/A
	SVD	r = [5,20,200]
	NMF	r = [5,20,200]
	UMAP	n_components = [5,20,200]
Clustering	K-Means	k = [10,20,50]
	Agglomerative Clustering	n_clusters = [20]
	HDBSCAN	min_cluster_size = [100,200]

The dimensionality technique that works best in this case is:-

Best Avg Score: 0.5548483341271374
Best Dimension: UMAP(metric='cosine', n_components=200)
Best Cluster: KMeans(max_iter=1000, n_clusters=20, n_init=30, random_state=0)

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.

XVIII. Question 18

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

We can try 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 can be used with UMAP(Cosine, r=20 for dimensionality reduction and K-means(k=20) forclustering.

- Try 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.

XIX. Question 19

If the VGG network is trained on a dataset with perhaps totally different classes as targets, why would one expect the features derived from such a network to have discriminative power for a custom dataset?

The VGG network, originally trained on the ImageNet dataset with diverse classes, is valuable for feature extraction in custom datasets due to its ability to capture fundamental image features in its lower layers. These features, such as edges and textures, are broadly applicable across various datasets. Although higher layers become more specific to ImageNet classes, they still encode abstract and generalized image representations, making them potentially discriminative for other datasets. The network's capacity to learn hierarchical and compositional features relevant to image structure facilitates their transferability, allowing the pre-trained VGG network to serve as an effective feature extractor for tasks like image clustering on different datasets, including the custom dataset in this project.

XX. Question 20

In a brief paragraph explain how the helper code base is performing feature extraction.

The provided helper code performs feature extraction using a pre-trained VGG network. First, it checks if a file containing pre-extracted features and labels exists; if not, it proceeds to download the flowers dataset, extract images, and define a FeatureExtractor class based on the VGG-16 architecture. This class is then used to initialize a feature extractor model. The dataset is loaded, and images are preprocessed using transformations such as resizing and normalization. The model iterates through the dataset, extracting features using the defined FeatureExtractor, and storing them along with corresponding labels. The extracted features are then reduced to 2D using Principal Component Analysis (PCA) for visualization. Additionally, the code includes the implementation of an MLP classifier and an autoencoder. The MLP classifier is designed with three fully connected layers and uses the negative log-likelihood loss for training. The autoencoder, designed to reduce the dimensionality to 2, is trained using mean squared error loss. Finally, the transformed features from the autoencoder are visualized in a scatter plot. Overall, this codebase efficiently performs feature extraction, classification, and dimensionality reduction for the flowers dataset.

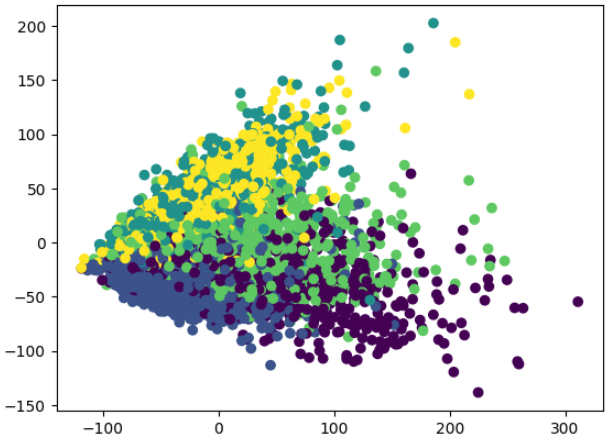


Figure 21: PCA Reduced

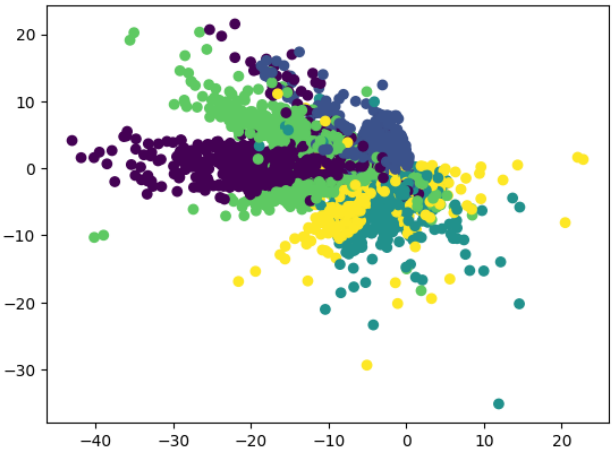


Figure 22: Autoencoder

XXI. Question 21

How many pixels are there in the original images? How many features does the VGG network extract per image; i.e what is the dimension of each feature vector for an image sample?

The original images are resized to 224x224 pixels during the preprocessing steps. Therefore, the original images have 224 * 224 = 50,176 pixels each.

The VGG network, specifically the VGG-16 architecture used in this code, extracts features from the images. The dimension of each feature vector extracted by the VGG network for an image sample is 4096. This is obtained from the flattening operation after passing the image through the convolutional layers, average pooling layer, and the first part of the fully connected layer in the VGG-16 architecture.

XXII. Question 22

Are the extracted features dense or sparse?

The extracted features from the VGG network are dense. In computer vision and deep learning, dense features mean that each element in the feature vector typically holds a non-zero value, and the majority of elements contribute to representing information about the input image. Dense features capture detailed information from the image and are commonly used in tasks such as image classification and clustering.

On the other hand, sparse features, as seen in TF-IDF (Term Frequency-Inverse Document Frequency) representations in text, have many zero values, indicating that only a small subset of features is relevant for a particular document. Sparse representations are beneficial in text data when dealing with a large number of unique terms, and each document only contains a small subset of those terms.

XXIII. Question 23

In order to inspect the high-dimensional features, t-SNE is a popular off-the-shelf choice for visualizing Vision features. Map the features you have extracted onto 2 dimensions with t-SNE. Then plot the mapped feature vectors along x and y axes. Color-code the data points with ground-truth labels. Describe your observation.

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.

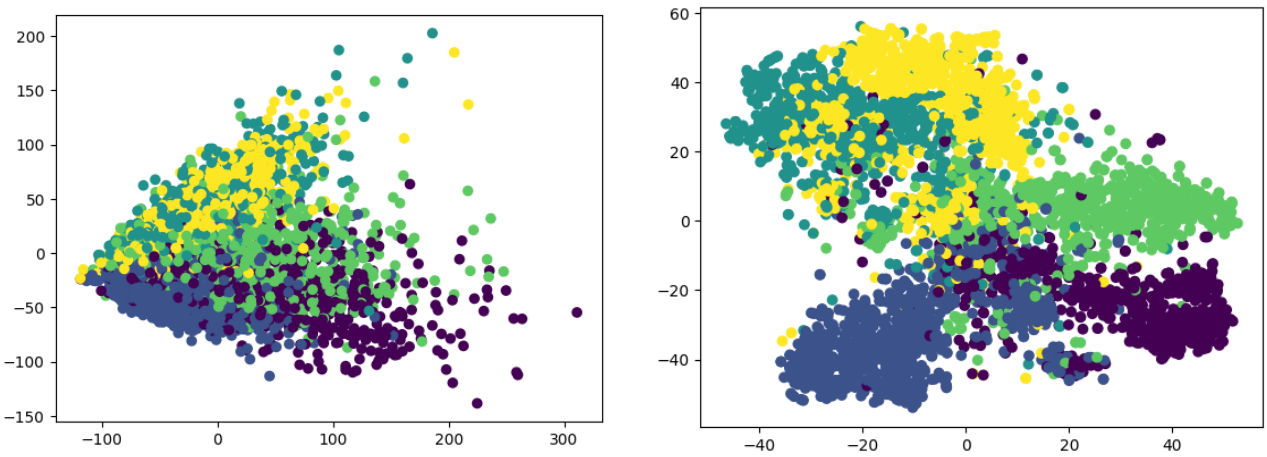


Figure 23: (Left) PCA (Right) TSNE

XXIV. Question 24

Report the best result (in terms of rand score) within the table below.
For HDBSCAN, introduce a conservative parameter grid over min cluster size and min samples.

Module	Alternatives	Hyperparameters
Dimensionality Reduction	None	N/A
	SVD	r = 50
	UMAP	n_components = 50
	Autoencoder	num_features = 50
Clustering	K-Means	k = 5
	Agglomerative Clustering	n_clusters = 5
	HDBSCAN	min_cluster_size & min_samples

The best results are as follows:-

Best Avg Score: 0.827863319081813
Best Dimension: UMAP(metric='cosine', n_components=50)
Best Cluster: AgglomerativeClustering(n_clusters=20)

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 asso- ciates clusters based on angle between sample points. Thus, cosine similarity is a much metric when compared to L2 distances. 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.

XXV. Question 25

Report the test accuracy of the MLP classifier on the original VGG features. Report the same when using the reduced-dimension features (you have freedom in choosing the dimensionality reduction algorithm and its parameters). Does the performance of the model suffer with the reduced-dimension representations? Is it significant? Does the success in classification make sense in the context of the clustering results obtained for the same features in Question 24.

Table : Performance Metric for Linear SVM Classifier with GLOVE embeddings

Dimension Reduction method	Test Accuracy
None	88.82
UMAP	86.64
Autoencoder	87.73
PCA	89.23

The optimal performance among all the mentioned combinations is observed with MLP using SVD. Even without dimensionality reduction, the test accuracy remains fairly close to that achieved with SVD. In clustering models, clusters are created based on distance, whereas MLP learns intricate features and makes predictions based on that acquired knowledge. As a result, MLP proves to be more effective than clustering models.

XXVI. Question 26

Try to construct various text queries regarding types of Pokémon (such as ”type: Bug”, ”electric type Pokémon” or ”Pokémon with fire abilities”) to find the relevant images from the dataset. Once you have found the most suitable template for queries, please find the top five most relevant Pokémon for type Bug, Fire and Grass. For each of the constructed query, please plot the five most relevant Pokémon horizontally in one figure with following specifications:

- the title of the figure should be the query you used;
- the title of each Pokémon should be the name of the Pokémon and its first and second type.

Repeat this process for Pokémon of Dark and Dragon types. Assess the effectiveness of your queries in these cases as well and try to explain any differences.

Tested out various queries for different types of Pokémon. Examples are as follows:-

For Fire Type:-

```
Query: type: Fire
Similarity Index for Magmar: 0.11063799262046814
--------------------------
Query: Pokemon with fire abilities
Similarity Index for Magmar: 0.08908902108669281
--------------------------
Query: fire type Pokemon

Similarity Index for Magmar: 0.1069665253162384
```

For Bug Type:-

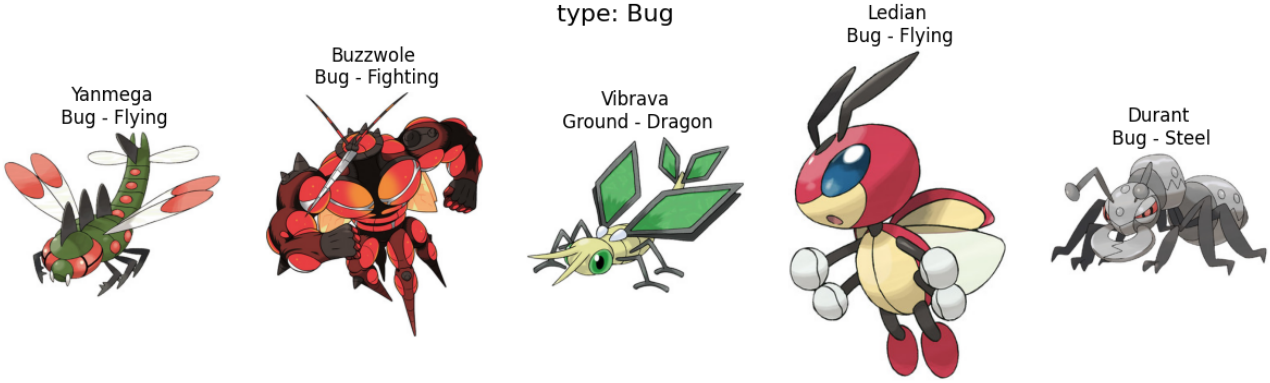
```
Query: type: bug
Similarity Index for Buzzwole: 0.05121926590800285
--------------------------
Query: Pokemon with bug abilities
Similarity Index for Buzzwole: 0.019698385149240494
--------------------------
Query: bug type Pokemon

Similarity Index for Buzzwole: 0.016686996445059776
```

We can see that the similarity index is higher for the first query as compared to the other queries.

Therefore, I used that query to find the top five most relevant Pokémon for each type.

We can see the relevant 5 pokemons below.

type: Bug



type: Fire

type: Grass

type: Dark

We can see that the queries are running pretty accurately from the output pokemons that we have received. The differences arises due to the model's understanding of textual descriptions and the complexity of the Pokemon dataset.

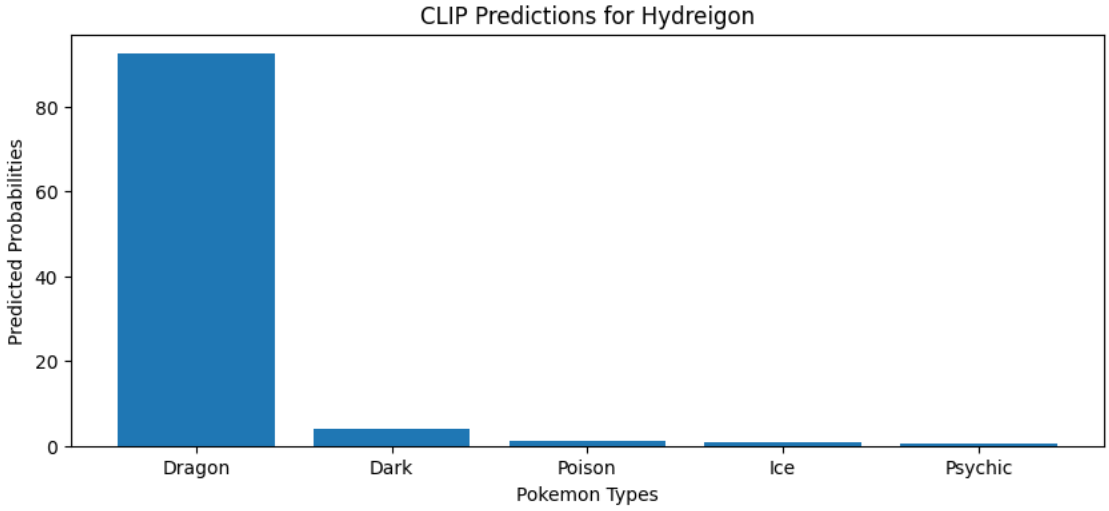
XXVII. Question 27

Randomly select 10 Pokémon images from the dataset and use CLIP to find the most relevant types (use your preferred template, e.g ”type: Bug”). For each selected Pokémon, please plot it and indicate:

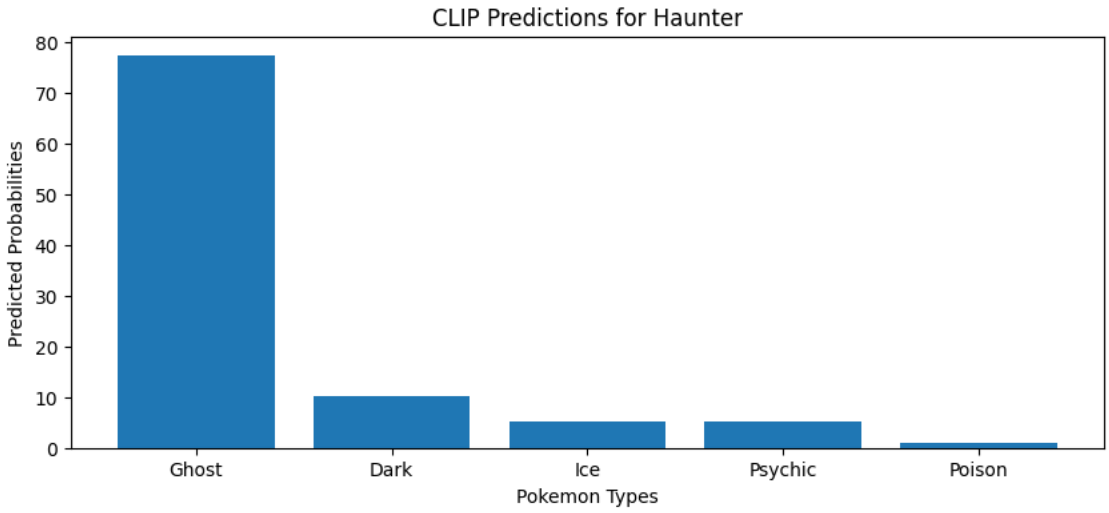
- its name and first and second type;
- the five most relevant types predicted by CLIP and their predicted probabilities.

I have plotted the most relevant types for each randomly selected Pokémon. For each plot, I have displayed the name and first and second type of the Pokémon.

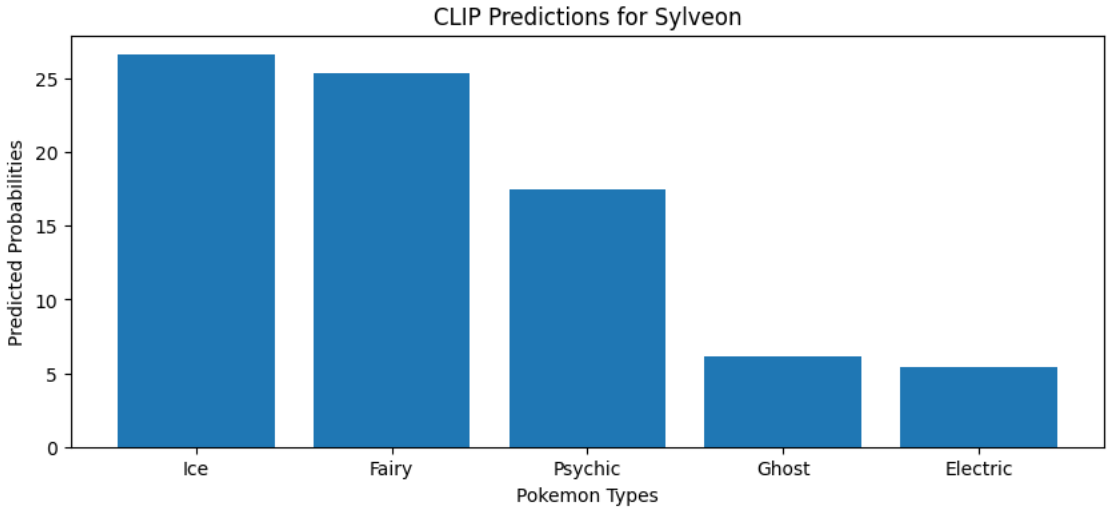
We can see the Pokémons and the corresponding relevant types above.



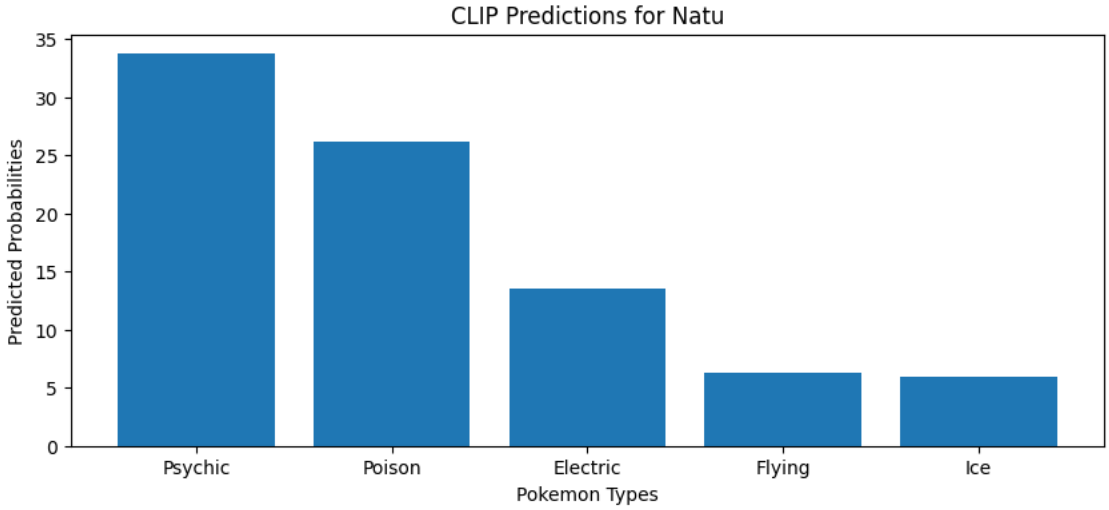
Haunter
Ghost - Poison



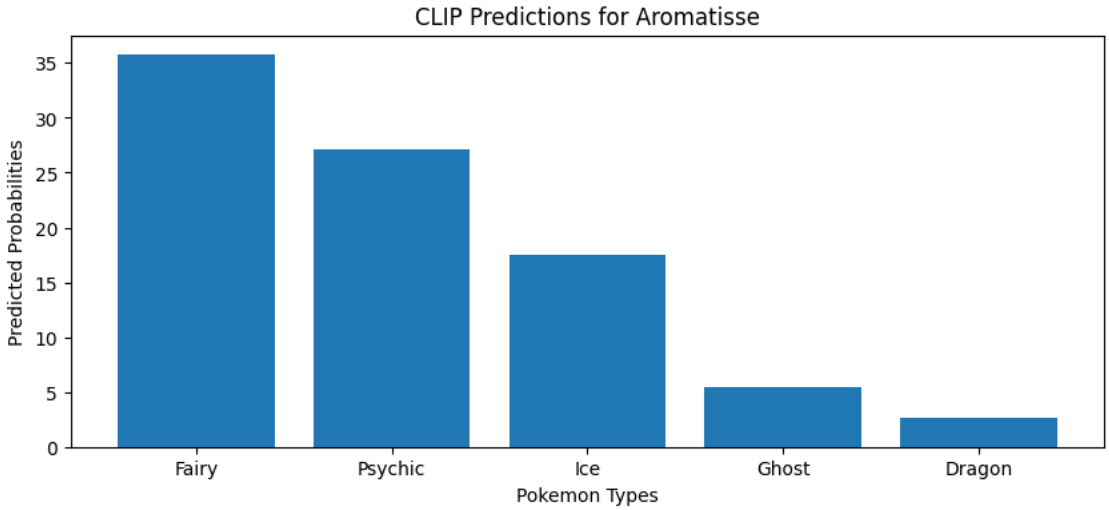
Sylveon
Fairy -



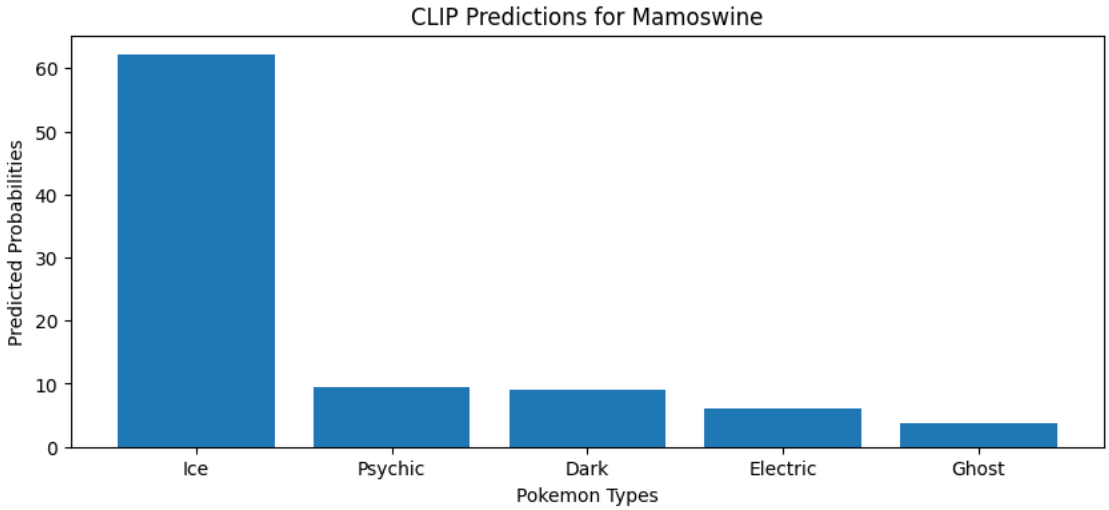
Natu
Psychic - Flying



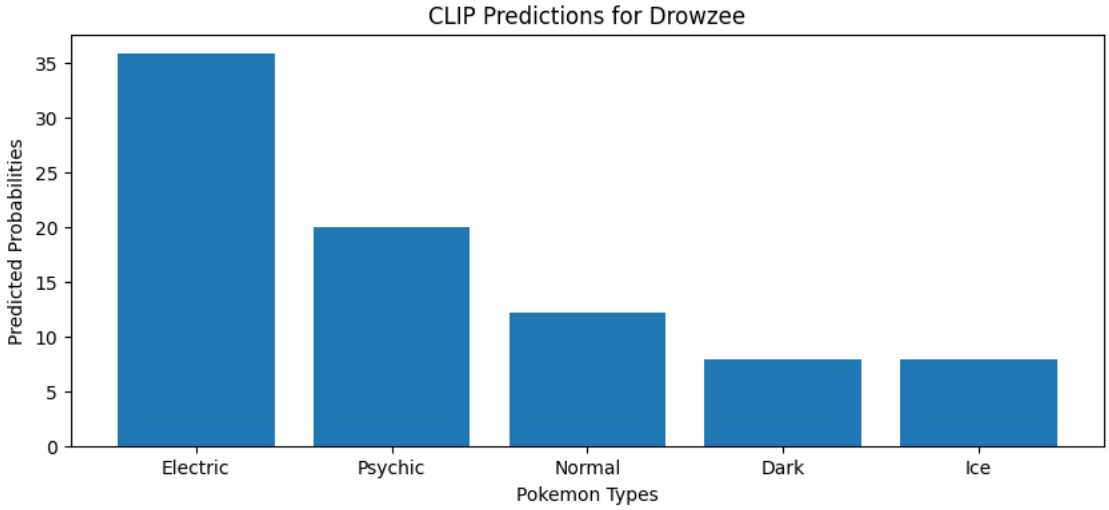
Aromatisse
Fairy -



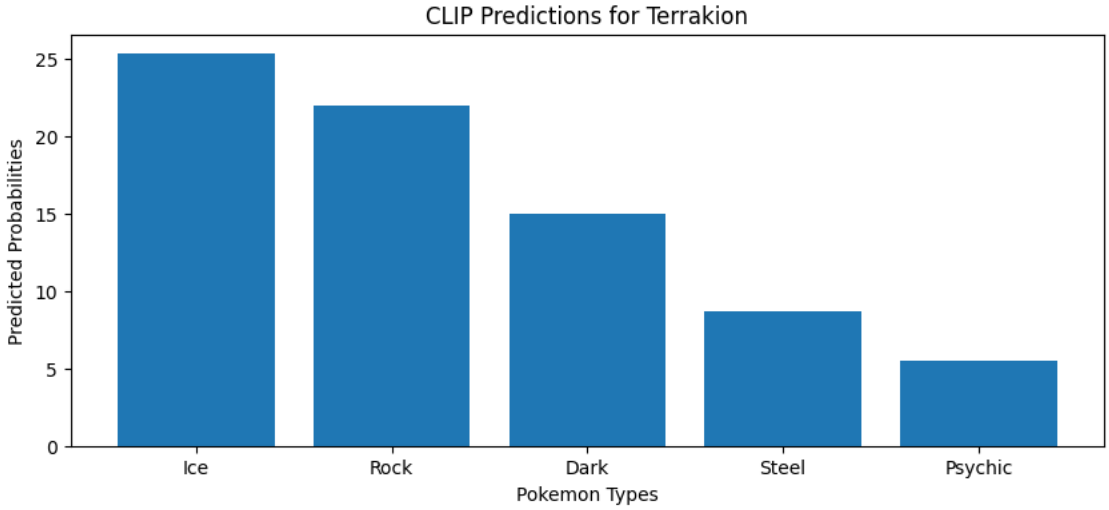
Mamoswine
Ice - Ground



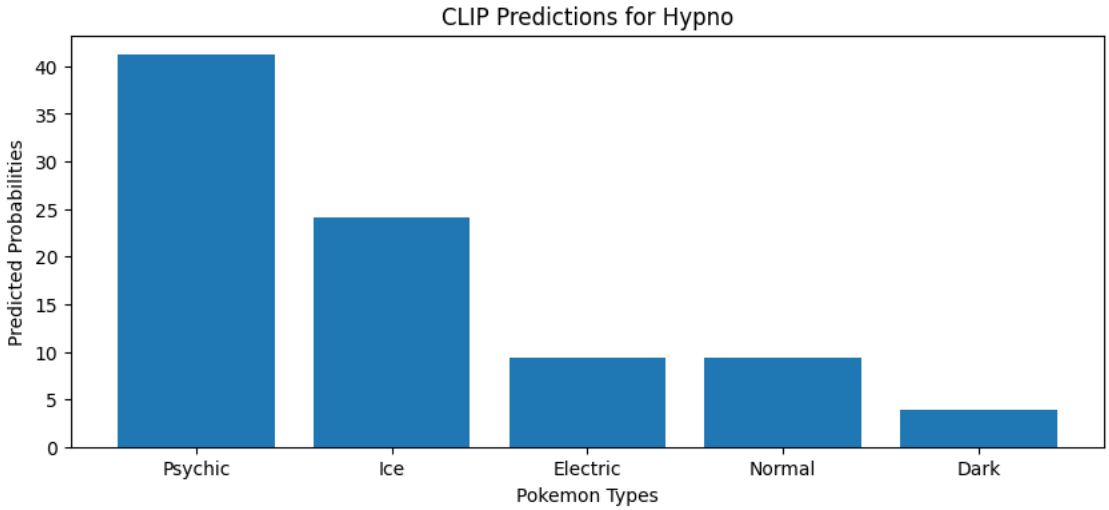
Drowzee
Psychic -



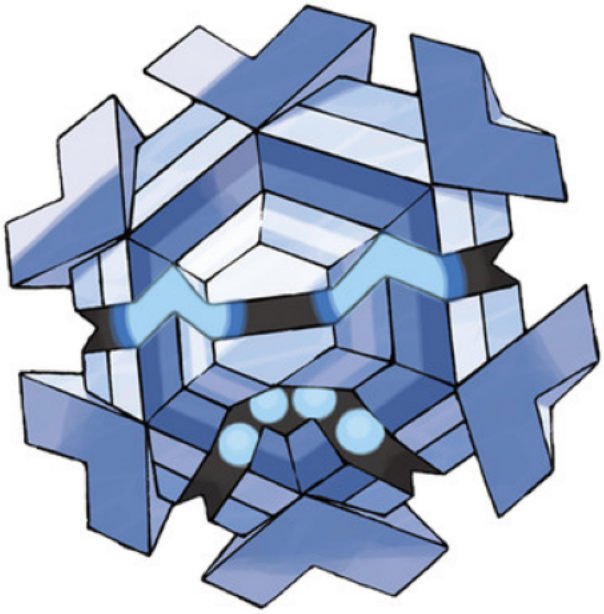
Terrakion
Rock - Fighting

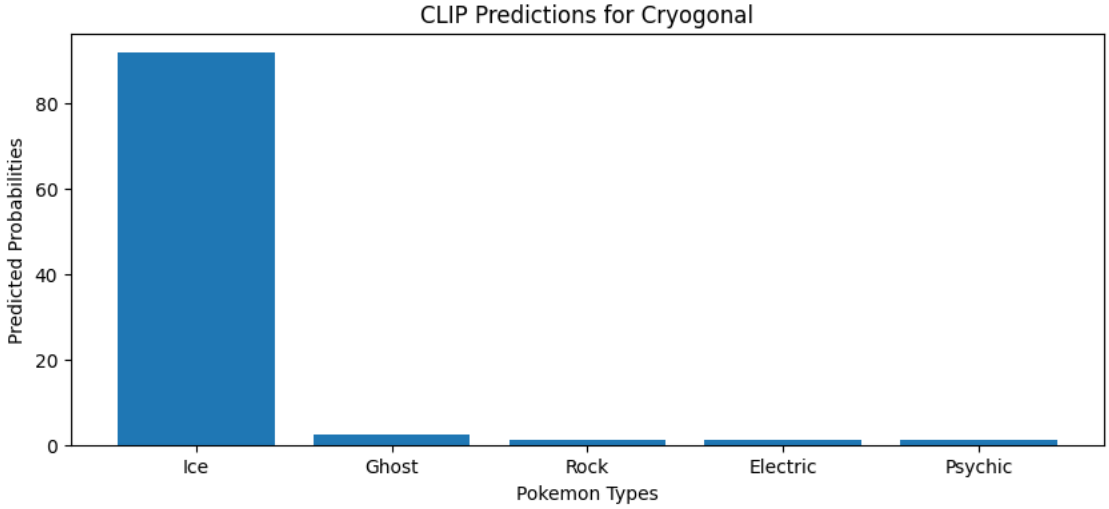


Hypno
Psychic -



Cryogonal
Ice -





XXVIII. Question 28

In the first and second question, we investigated how CLIP creates 'clusters' by mapping images and texts of various Pokémon into a high-dimensional space and explored neighbor- hood of these items in this space. For this question, please use t-SNE to visualize image clusters, specifically for Pokémon types Bug, Fire, and Grass. You can use scatter plot from python package plotly. For the visualization, color-code each point based on its first type type 1 using the 'color' argument, and label each point with the Pokémon's name and types using 'hover name'. This will enable you to identify each Pokemon represented in your visualization. After completing the visualization, analyze it and discuss whether the clustering of Pokémon types make sense to you.

We can see that:-

- Pokémon of the same type tend to be clustered together. For example, there is a cluster of fire-type Pokémon in the left corner of the image, and a cluster of grass-type Pokémon in the bottom corner.
- There are some exceptions to this pattern. For example, there are a few fire-type Pokémon that are clustered with bug-type Pokémon. This may be because these Pokémon have some characteristics that are more similar to bug-type Pokémon than to other fire-type Pokémon.
- Some types are more spread out than others. For example, the grass-type Pokémon are more spread out than the fire-type Pokémon. This may be because grass-type Pokémon are more diverse than fire-type Pokémon.