UCLA

ECE 219 - Large-Scale Data Mining: Models and Algorithms

Winter 2024

Project 2: Data Representation and Clustering

Sunday, February 11, 2024

Author:

Gian Zignago (UID: 706294998)

QUESTION 1. Report the dimensions of the TF-IDF matrix you obtain.

The dimensions of the obtained TF-IDF matrix are **(7882, 18469)**

QUESTION 2. Report the contingency table of your clustering result. Does the contingency matrix have to be square-shaped?

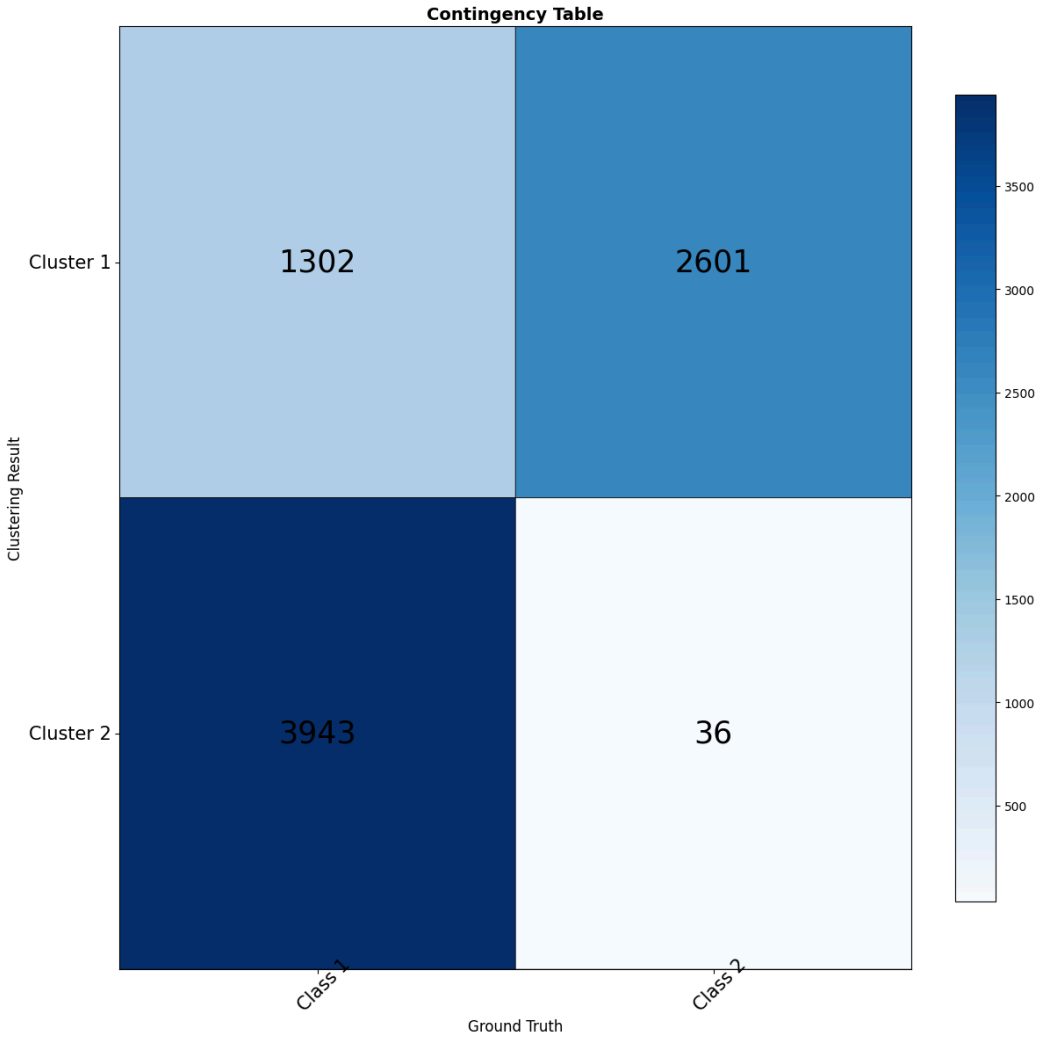


Figure 1: Contingency Table of Clustering Result

The contingency matrix **does not** have to be square-shaped. The shape of the matrix depends on the number of classes and the number of clusters (In this case two, since k=2 for K-means clustering). If there are more or less than two classes in the ground truth labels, the matrix will be rectangular.

QUESTION 3. Report the 5 clustering measures explained in the introduction for K-means clustering.

Homogeneity	0.427
Completeness	0.464
V-measure	0.445
Adjusted Rand Index	0.436
Silhouette Coefficient	0.009

Table 1: Five Described Measures for K-Means Clustering

The purpose of these measures is to assess the performance of a given clustering technique. Scores closer to 1.000 in each category signify higher performance. As can be seen in Table 1, the initial results garnered from our default K-means clustering algorithm are middling. This means that there is room for improvement in this clustering technique.

QUESTION 4. Report the plot of the percentage of variance that the top r principal components retain v.s. r, for r = 1 to 1000.

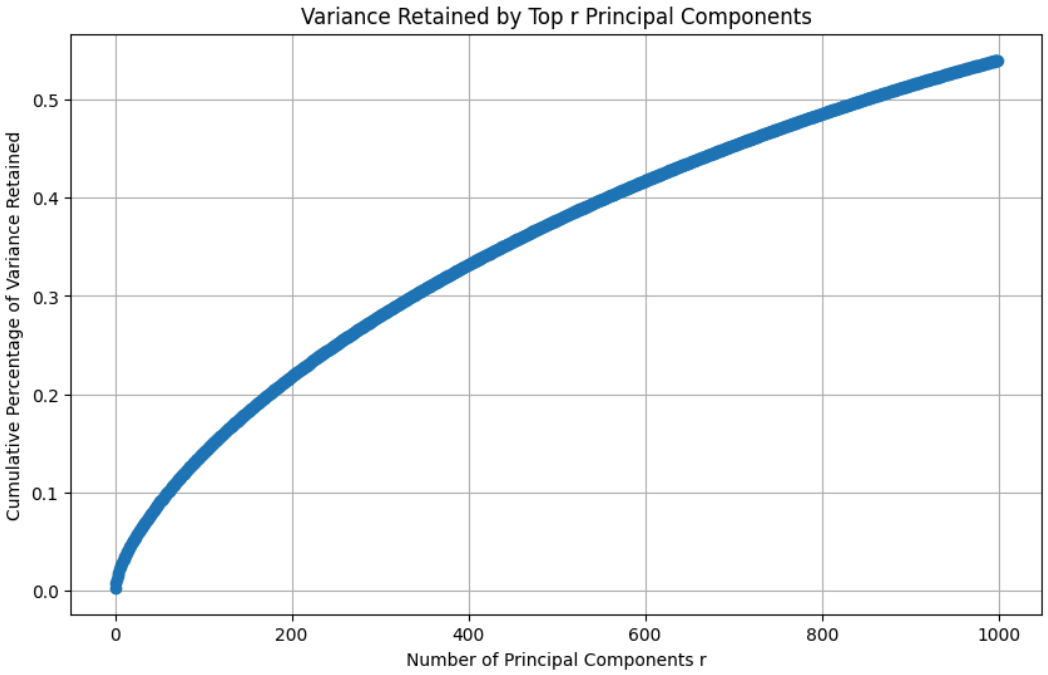


Figure 2: Variance Retained by Top *r* Principal Components for *r* = 1 to *r* = 1000

QUESTION 5. Let r be the dimension that we want to reduce the data to (i.e. n components). Try r = 1 − 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. Note: In the choice of r, there is a trade-off between the information preservation, and better performance of k-means in lower dimensions.

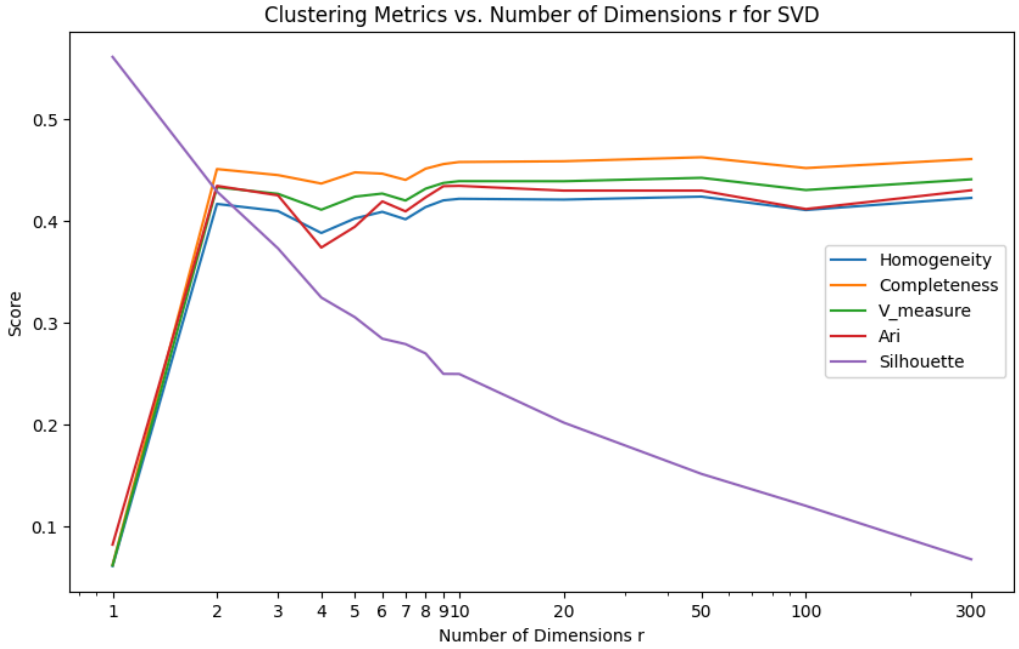


Figure 3: Relevant Measure Scores Plotted Over Variable *r* for SVD

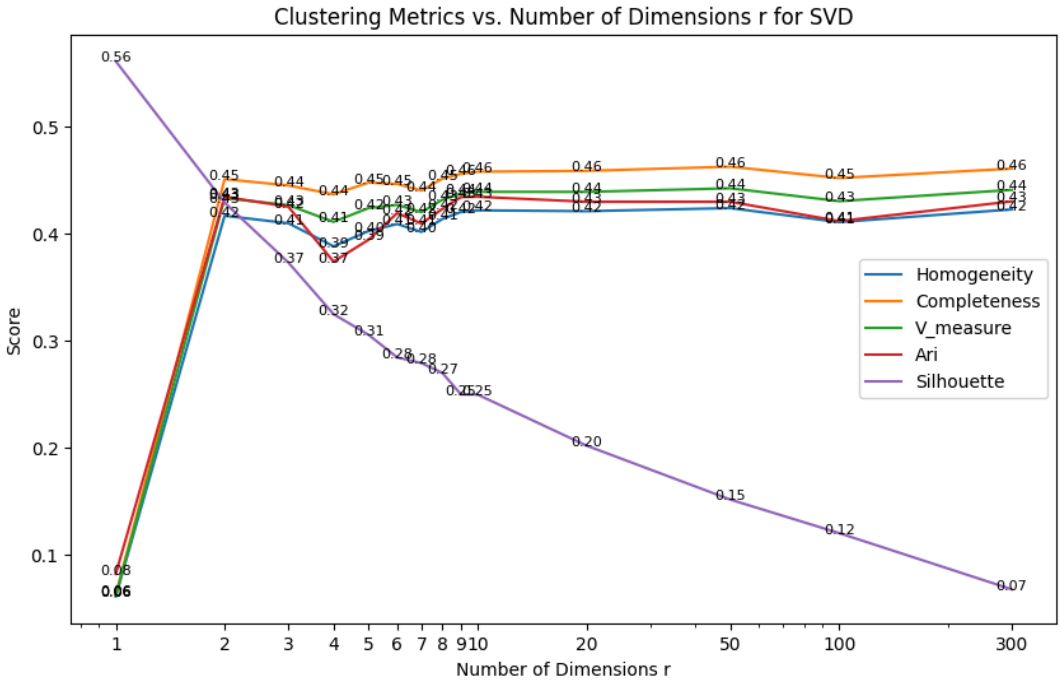


Figure 4: SVD Plot with Scores Shown at Each Data Point

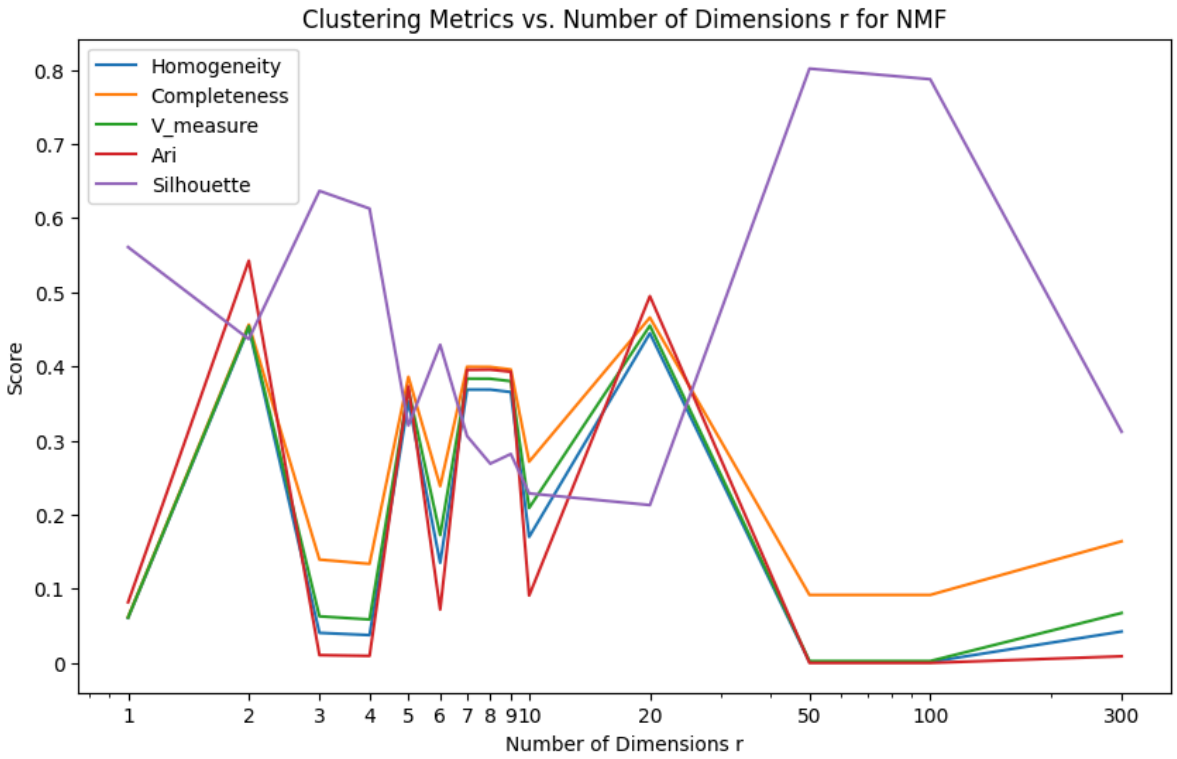


Figure 5: Relevant Measure Scores Plotted Over Variable *r* for NMF

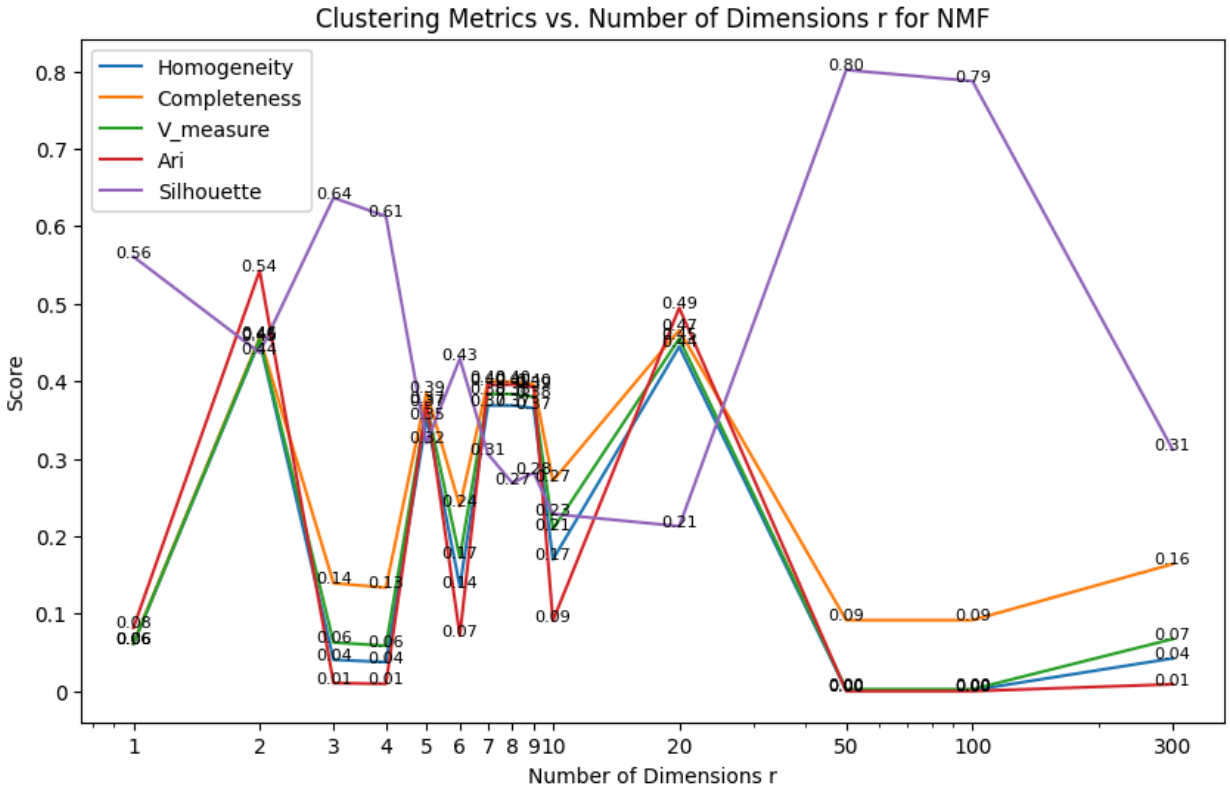


Figure 6: NMF Plot with Scores Shown at Each Data Point

For SVD and NMF: The "good" choice of *r* is a balance between dimensionality (lower is generally better for K-means) and the retention of meaningful variance in the data (higher *r* retains more information). Typically, you would look for an elbow in the plot of each metric versus *r* or a point where increases in r yield diminishing returns in metric improvement.

From Figures 4 and 5, we can conclude that a strong choice of *r* for SVD is 2, as it is the only value that retains scores between 0.42 and 0.45 for each of the 5 relevant variables. While *r* values in the 10-50 range show marginal improvements (score values 0.01 higher) in some variables, the performance of the Silhouette variable plummets as r increases, making higher *r* values less appealing.

While the NMF plot shown in Figures 5 and 6 are less consistent than the plots for SVD, a good candidate of *r* for NMF is 2. For NMF, 2 is one of the few *r* values with consistent scores for all 5 relevant variables, and the only one among those to have all 5 variables score above 0.40.

QUESTION 6. How do you explain the non-monotonic behavior of the measures as r increases?

The non-monotonic behavior of the clustering measures as the number of dimensions *r* increases can be explained by the complexity of the data structure and the nature of each dimensionality reduction technique. As well as this, the specific characteristics of the clustering algorithms should be taken into account.

For both NMF and SVD, Noise and Signal Separation is one cause of this non-monotonic behavior. Initially, as *r* increases from 1, more signal is captured. This leads to an improvement in clustering performance, as indicated by the initial rise in the score of some metrics. However, after a certain point, adding more dimensions introduces noise and less informative features, degrading the clustering performance and causing the scores to decrease. With a higher number of dimensions, the clustering algorithms might also start to overfit to the noise in the data instead of identifying true clusters, leading to further fluctuations in the performance metrics.

NMF specifically can lead to denser representations as *r* increases, which can sometimes deteriorate the performance of clustering algorithms that benefit from sparse data representations. Some dimensions may also align well with the underlying cluster structure, while others do not, resulting in non-monotonic changes in the metrics.

QUESTION 7. Are these measures on average better than those computed in Question 3?

For SVD, the measures on average resemble those computed through K-Means Clustering in Question 3, especially as r increases. From Figure 4, it can be inferred that an r value between 100 and 300 would resemble the metrics found in Question 3. It should be noted that performance is significantly worse for all metrics except for Silhouette when $r = 1$. For lower r values, however, the rest of the metrics have similar values to the results in Question 3 except for the Silhouette value, which is significantly higher than the value of 0.09 calculated using K-Means Clustering. Thus, it can be inferred that SVD would perform similarly, if not slightly better, compared to K-Means Clustering.

For NMF, most values of r shown in Figure 6 perform worse than the results in Question 3. Two notable outliers, however, are when $r = 2$ and when $r = 20$. In these two specific scenarios, NMF performs better than the K-Means Clustering result in Question 3. On average, however, the measures for NMF perform worse than those computed in Question 3.

When combining the results for SVD and NMF, it is apparent that in aggregate, these measures on average perform slightly worse than those computed in Question 3.

QUESTION 8. Visualize the clustering results for SVD with your optimal choice of r for K-Means clustering and NMF with your choice of r for K-Means clustering.

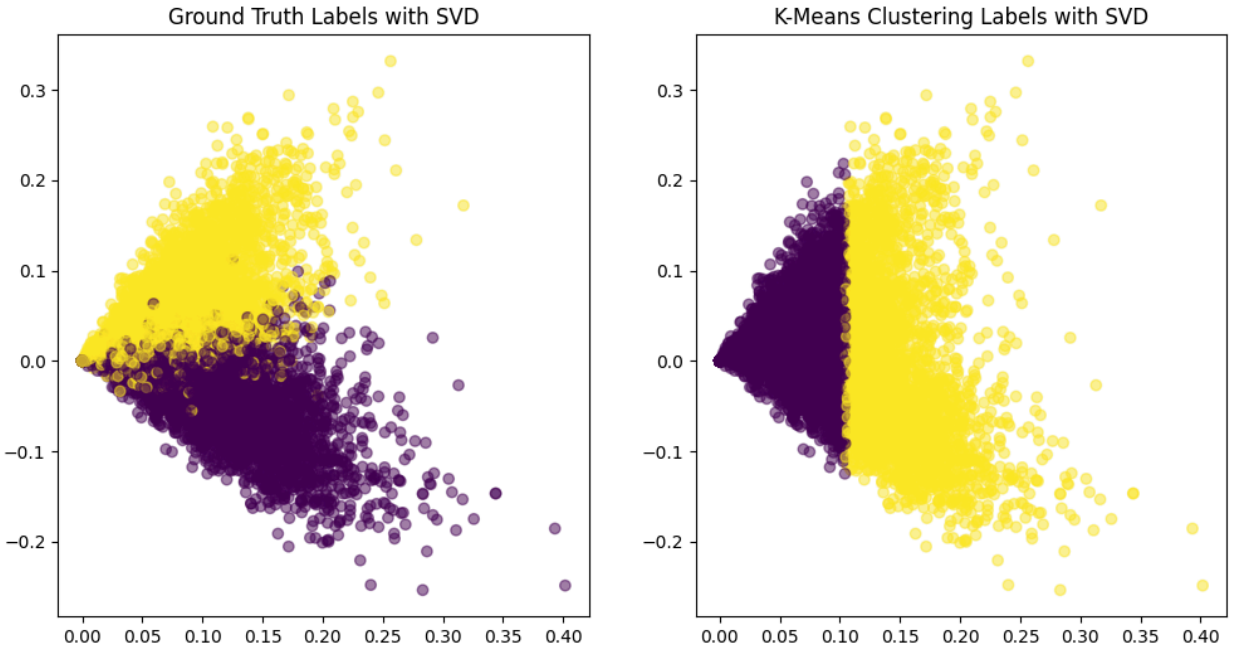


Figure 7: Clustering Results for SVD With $r = 1$

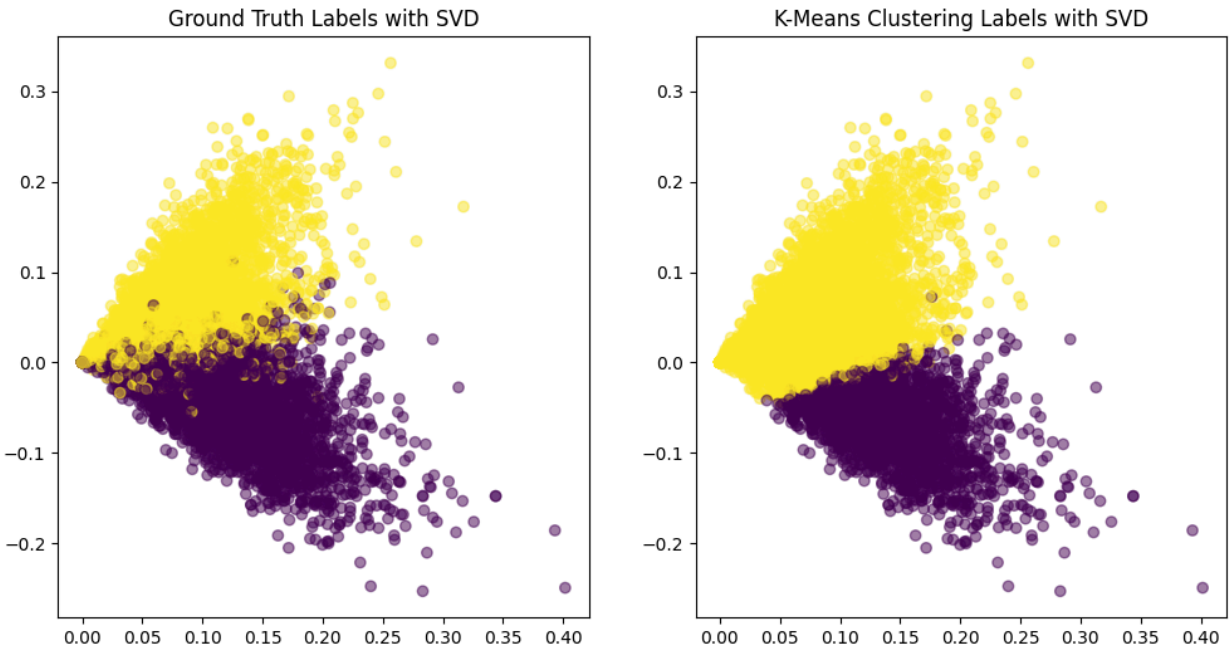


Figure 8: Clustering Results for SVD With $r = 2$

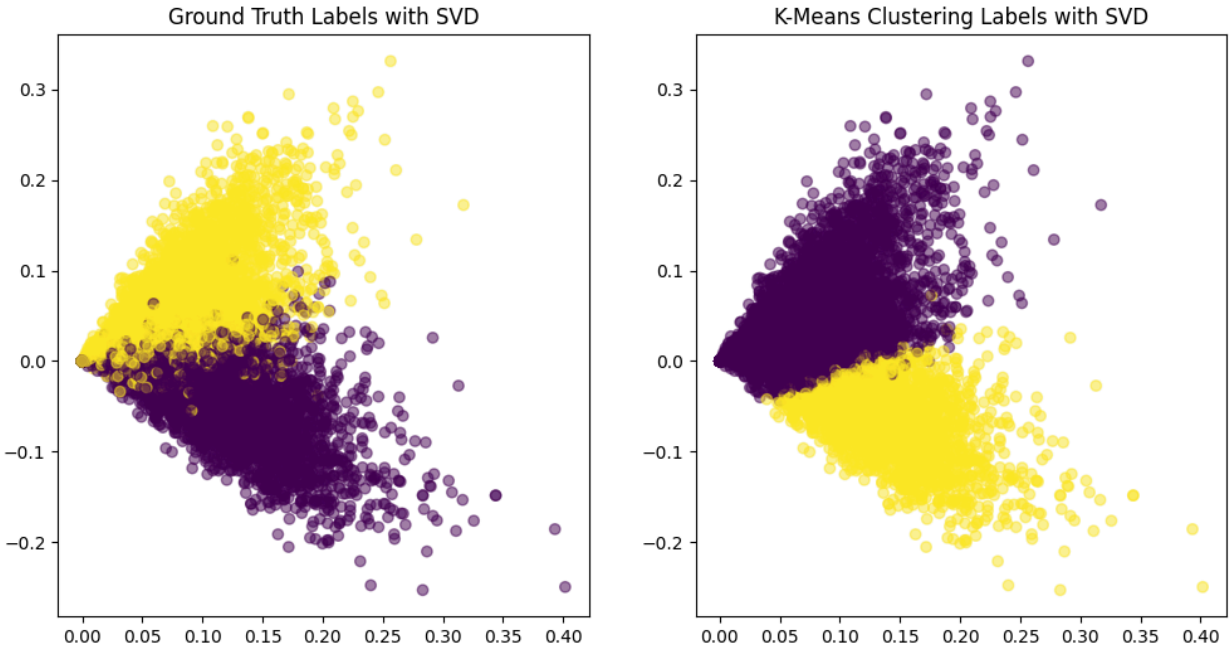


Figure 9: Clustering Results for SVD With $r = 100$

Figure 10: Clustering Results for SVD With $r = 300$

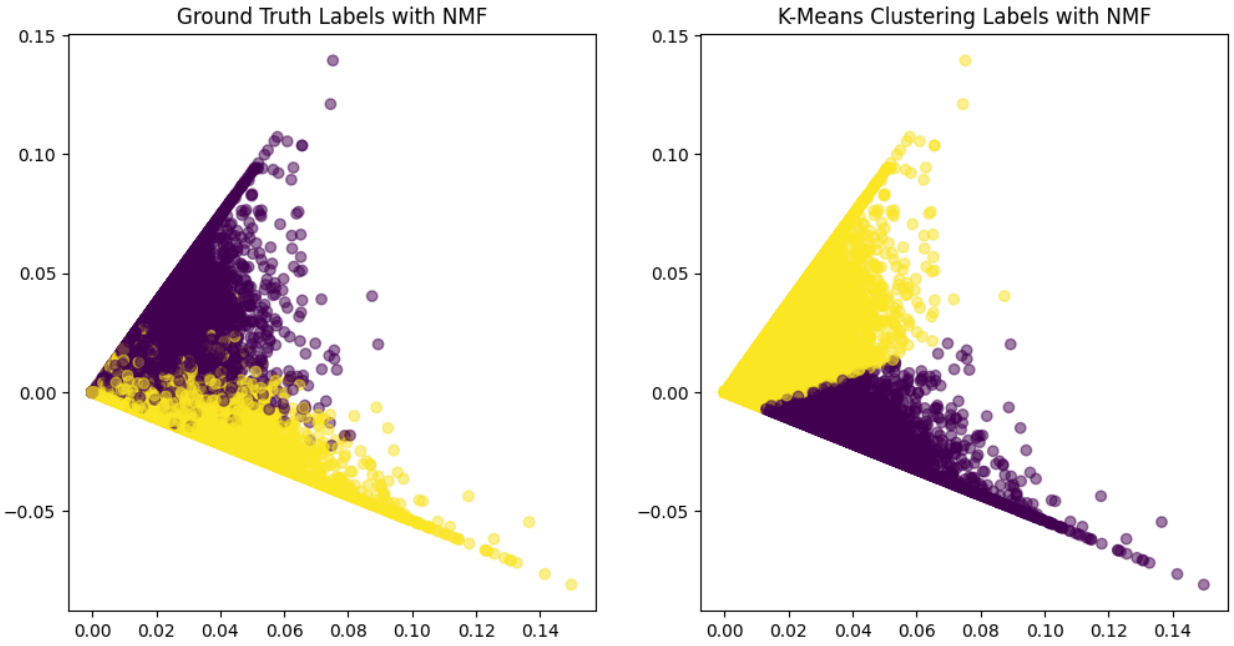


Figure 11: Clustering Results for NMF With $r = 2$

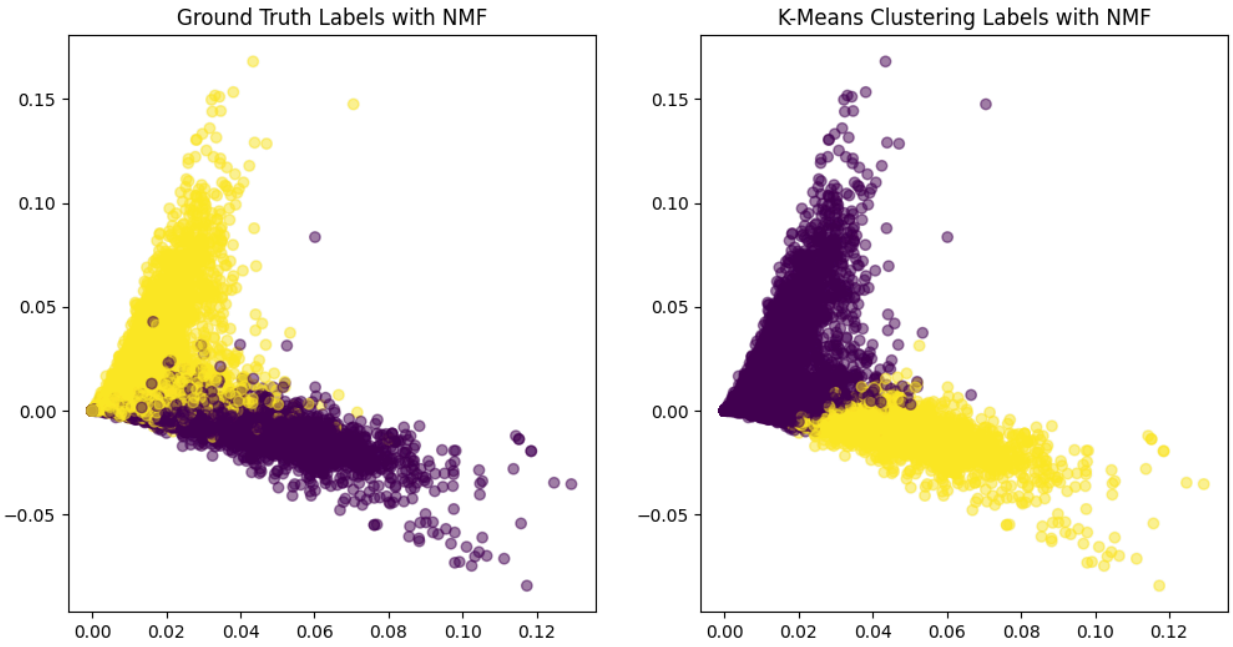


Figure 12: Clustering Results for NMF With $r = 20$

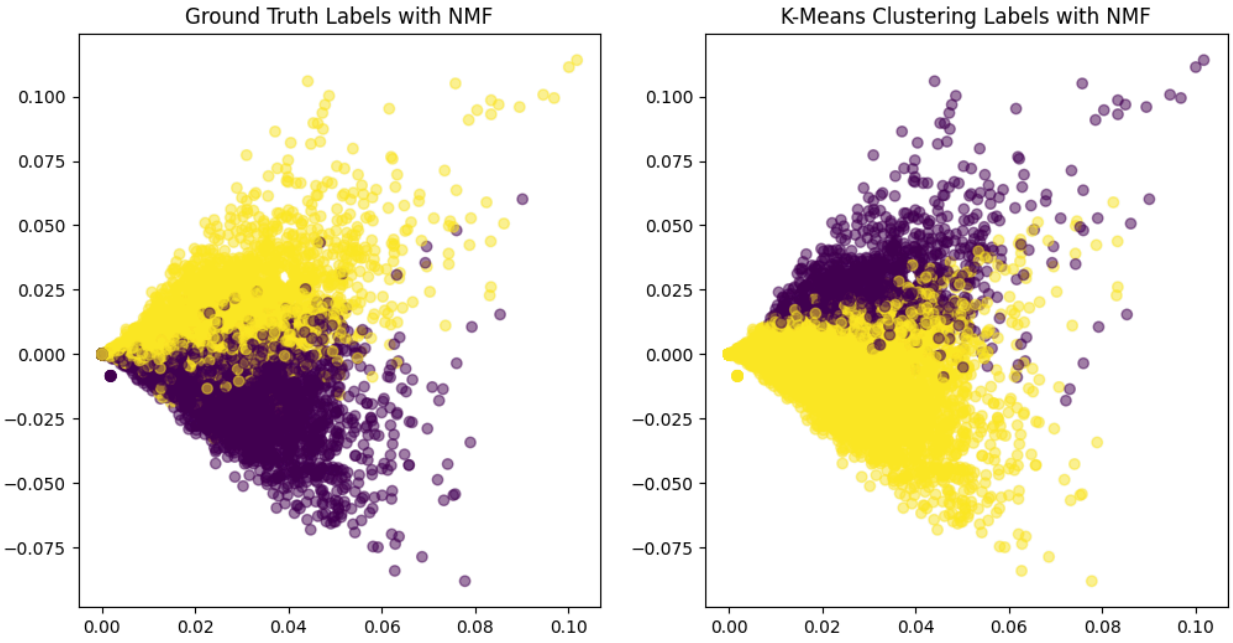


Figure 13: Clustering Results for NMF With $r = 50$

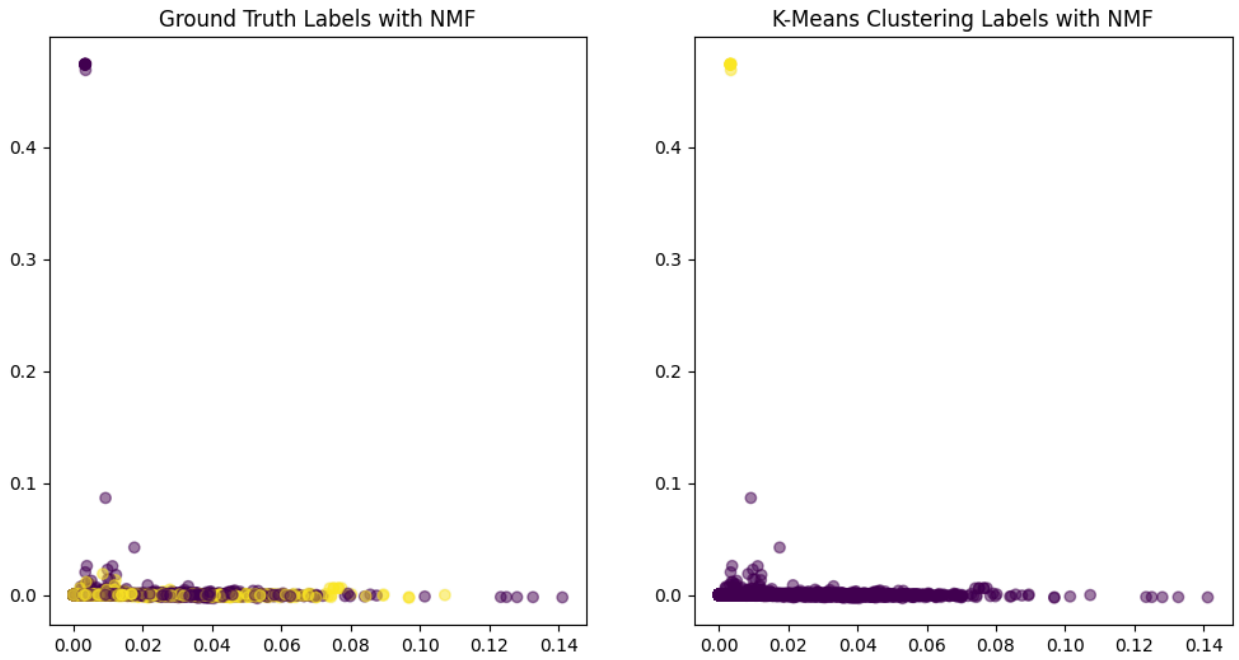


Figure 14: Clustering Results for NMF With $r = 100$

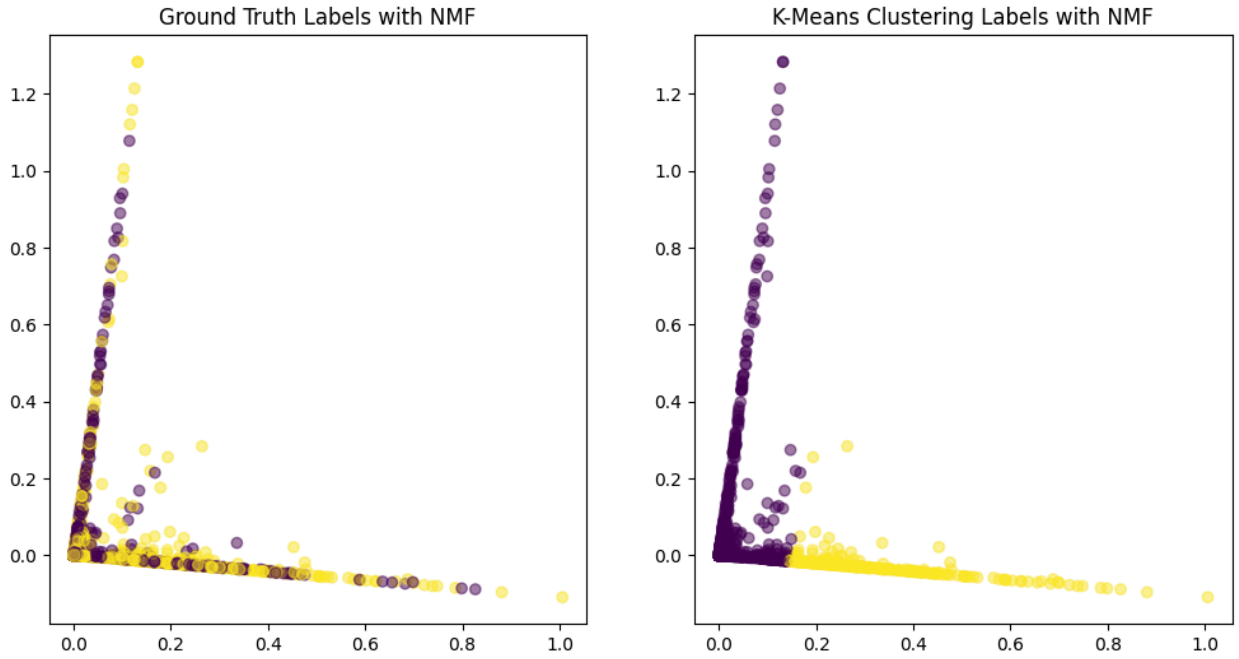


Figure 15: Clustering Results for NMF With $r = 300$

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?

General observations

The color differentiation between the ground truth labels and the clustering labels can tell us how well the K-Means algorithm was able to reconstruct the inherent grouping of the dataset. If colors in the ground truth and clustering labels broadly align, K-Means has performed well. Dense areas of points suggest higher concentrations of data, which K-Means favors due to its reliance on the mean as the cluster center. Sparse areas or outliers can skew the clustering results. Significant overlap between points of different colors in the ground truth visualization indicates that classes are not distinctly separable, a challenge for K-Means clustering.

Data point distribution using SVD

While SVD has identified underlying structures in the data, the distribution is not entirely ideal for K-Means clustering due to the non-spherical nature of the clusters and the variance in density.

Data point distribution using NMF

From the figures, we can see that the clusters are non-spherical and elongated in shape, which is not ideal for K-Means clustering. The algorithm prefers clusters to be round and of similar size. The distribution of points does not form clear separable clusters. Instead, there is a significant overlap between different classes. This is likely because the dataset is too complex for K-Means to effectively discern distinct groups. The NMF reduction produces a triangular structure for higher r values, with points distributed along the edges and less so in the center. This does not align with the K-Means assumption of cluster distribution.

QUESTION 10. Load documents with the same configuration as in Question 1, but for ALL 20 categories. Construct the TF-IDF matrix, reduce its dimensionality using BOTH NMF and SVD (specify settings you choose and why), and perform K-Means clustering with k=20 . Visualize the contingency matrix and report the five clustering metrics (DO BOTH NMF AND SVD).

Chosen Settings and Why:

```
X_reduced_svd =
TruncatedSVD(n_components=100).fit_transform(tfidf_matrix_all)
X_reduced_nmf = NMF(n_components=100).fit_transform(tfidf_matrix_all)
kmeans_cluster = KMeans(n_clusters=20, random_state=0, n_init=30,
max_iter=1000)
```

The choice of *n_components* for both NMF and SVD is driven by the desire to reduce the dimensionality of the TF-IDF matrix to a level that retains significant structure of the data and features while making the dataset manageable for clustering algorithms. Too few dimensions might lose important information, while too many dimensions could lead to the "curse of dimensionality," where distances become less meaningful. For this exercise, we've chosen *n_components* = 100. This choice is a balance between complexity and manageability, aiming to capture the core topics or variations in the dataset while avoiding overly sparse or high-dimensional spaces that could hinder clustering performance.

For SVD, *n_components* = 100 is selected to capture the most significant directions of variance in the data. SVD decomposes the TF-IDF matrix into three matrices, where the singular values reflect the importance of each new dimension in capturing the variance of the data. For NMF, *n_components* = 100 is also selected. The rationale is similar to SVD in terms of dimensionality, but the interpretation is slightly different. NMF components represent topics, with each document being a mixture of these topics. The non-negativity constraint makes these components more interpretable.

Homogeneity	0.2773792882151879
Completeness	0.37282617685283154
V-measure	0.3180971711845264
Adjusted Rand Index	0.05992137354414444
Silhouette Coefficient	0.0069727401561086215

Table 2: Five Clustering Metrics After Reducing Dimensionality with SVD

Homogeneity	0.09248923986722311
Completeness	0.1739703048029724
V-measure	0.12077166363555526
Adjusted Rand Index	0.006838404461430072
Silhouette Coefficient	0.008020431239131317

Table 3: Five Clustering Metrics After Reducing Dimensionality with NMF

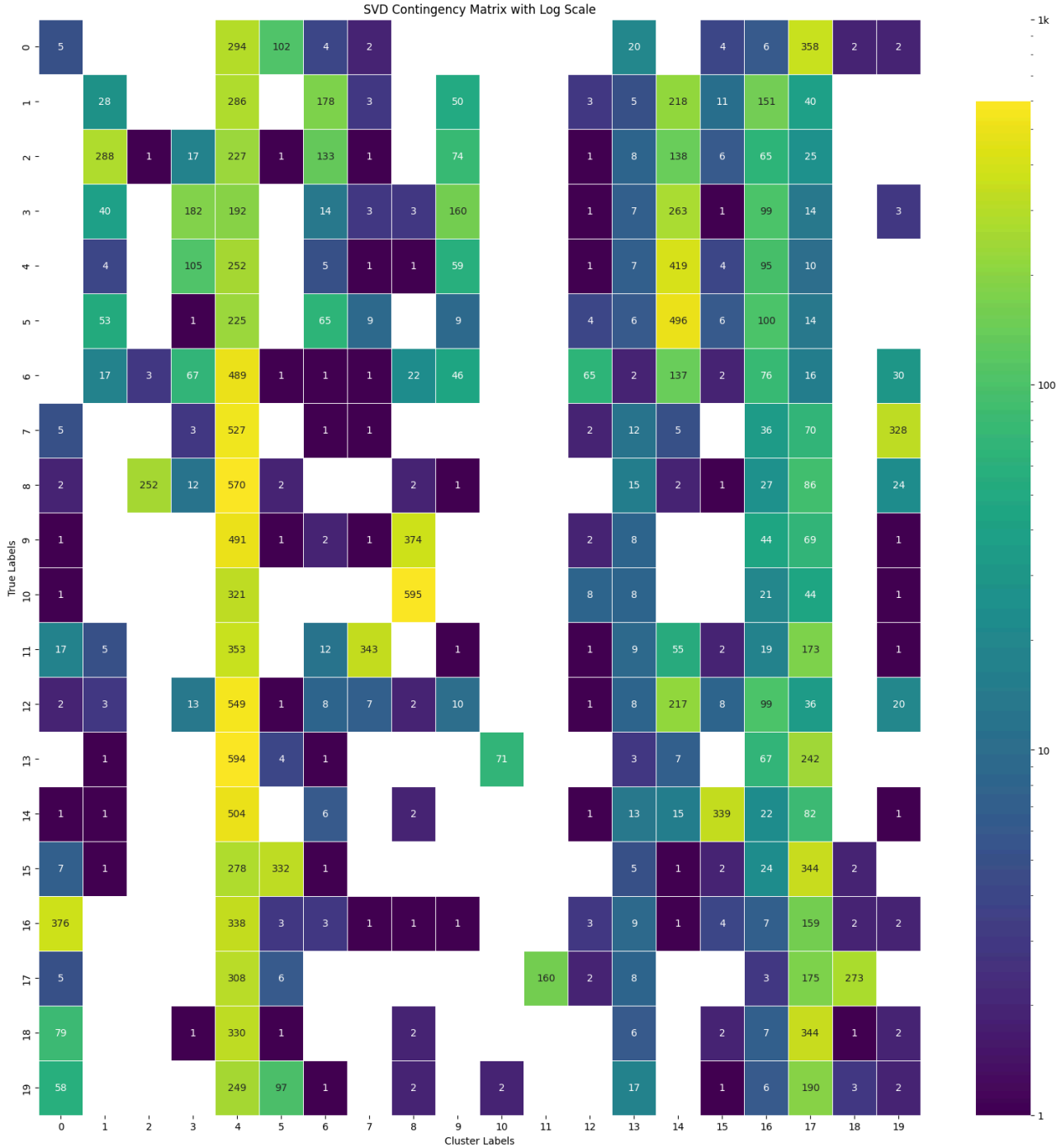


Figure 16: Contingency Matrix for SVD with a Logarithmic Scale

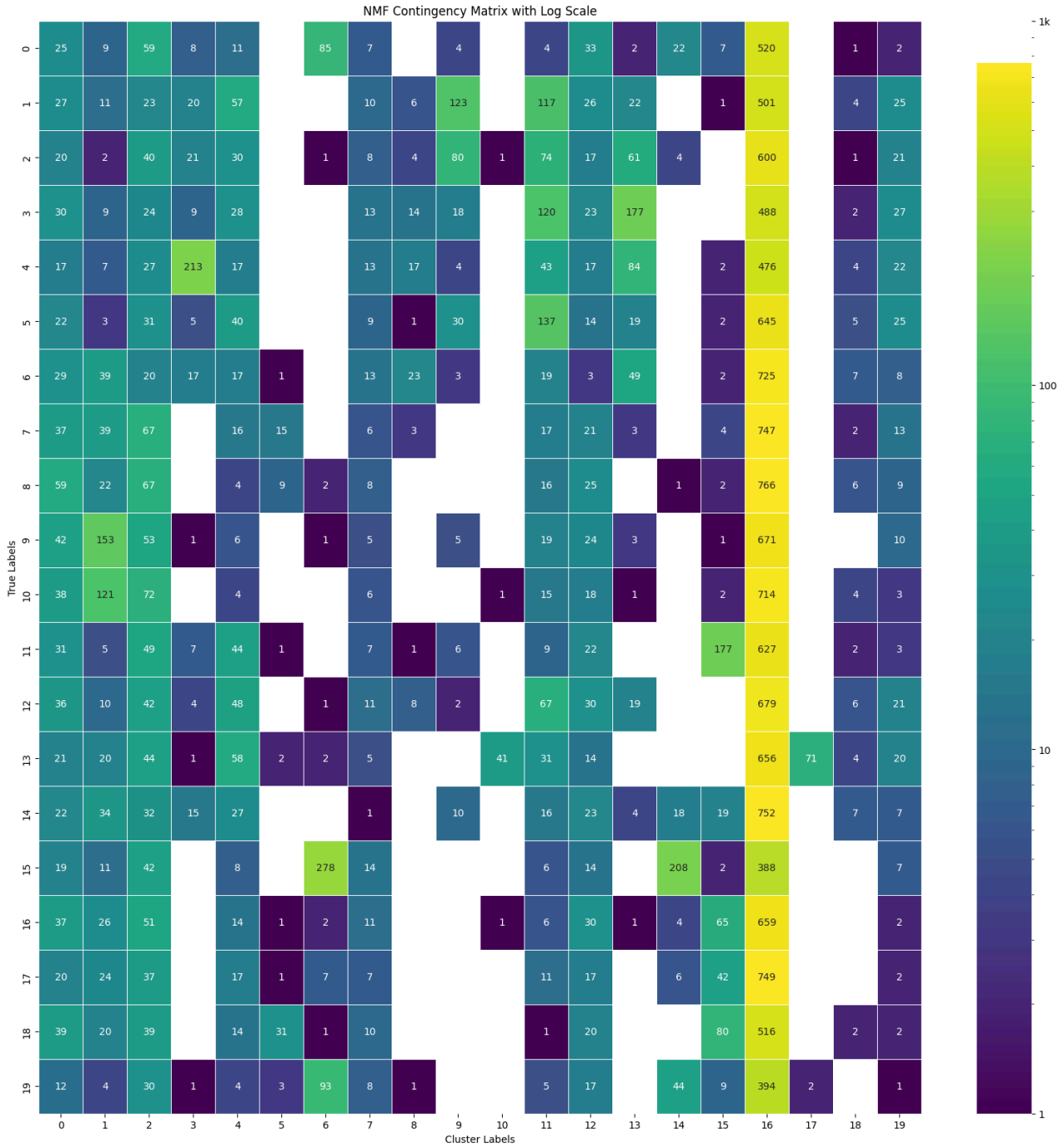


Figure 17: Contingency Matrix for NMF with a Logarithmic Scale

QUESTION 11. Reduce the dimension of your dataset with UMAP. Consider the following settings: n components = [5, 20, 200], metric = ”cosine” vs. ”euclidean”. Report the permuted contingency matrix and the five clustering evaluation metrics for the different combinations (6 combinations).

Homogeneity	0.45780407654707433
Completeness	0.47570566454397306
V-measure	0.466583224316939
Adjusted Rand Index	0.31894719851726877
Silhouette Coefficient	0.3641633987426758

Table 4: Results for n_components = 5, metric = cosine

Homogeneity	0.008625221377907765
Completeness	0.009557893776668271
V-measure	0.009067637643986853
Adjusted Rand Index	0.0006061746129830106
Silhouette Coefficient	0.17847539484500885

Table 5: Results for n_components = 5, metric = euclidean

Homogeneity	0.4507950604868615
Completeness	0.4713194543397386
V-measure	0.46082884177931166
Adjusted Rand Index	0.309741181018409
Silhouette Coefficient	0.3650643229484558

Table 6: Results for n_components = 20, metric = cosine

Homogeneity	0.009689000933811564
Completeness	0.010898578978627867
V-measure	0.010258256905401596
Adjusted Rand Index	0.0007811873467745634
Silhouette Coefficient	0.12074040621519089

Table 7: Results for n_components = 20, metric = euclidean

Homogeneity	0.4542294271439196
Completeness	0.47048568004834757
V-measure	0.46221466323107013
Adjusted Rand Index	0.3158798436325307
Silhouette Coefficient	0.36254817247390747

Table 8: Results for n_components = 200, metric = cosine

Homogeneity	0.007446659453008432
Completeness	0.009300625164774248
V-measure	0.008271023020485313
Adjusted Rand Index	0.0003930937670395822
Silhouette Coefficient	0.16099224984645844

Table 9: Results for n_components = 200, metric = euclidean

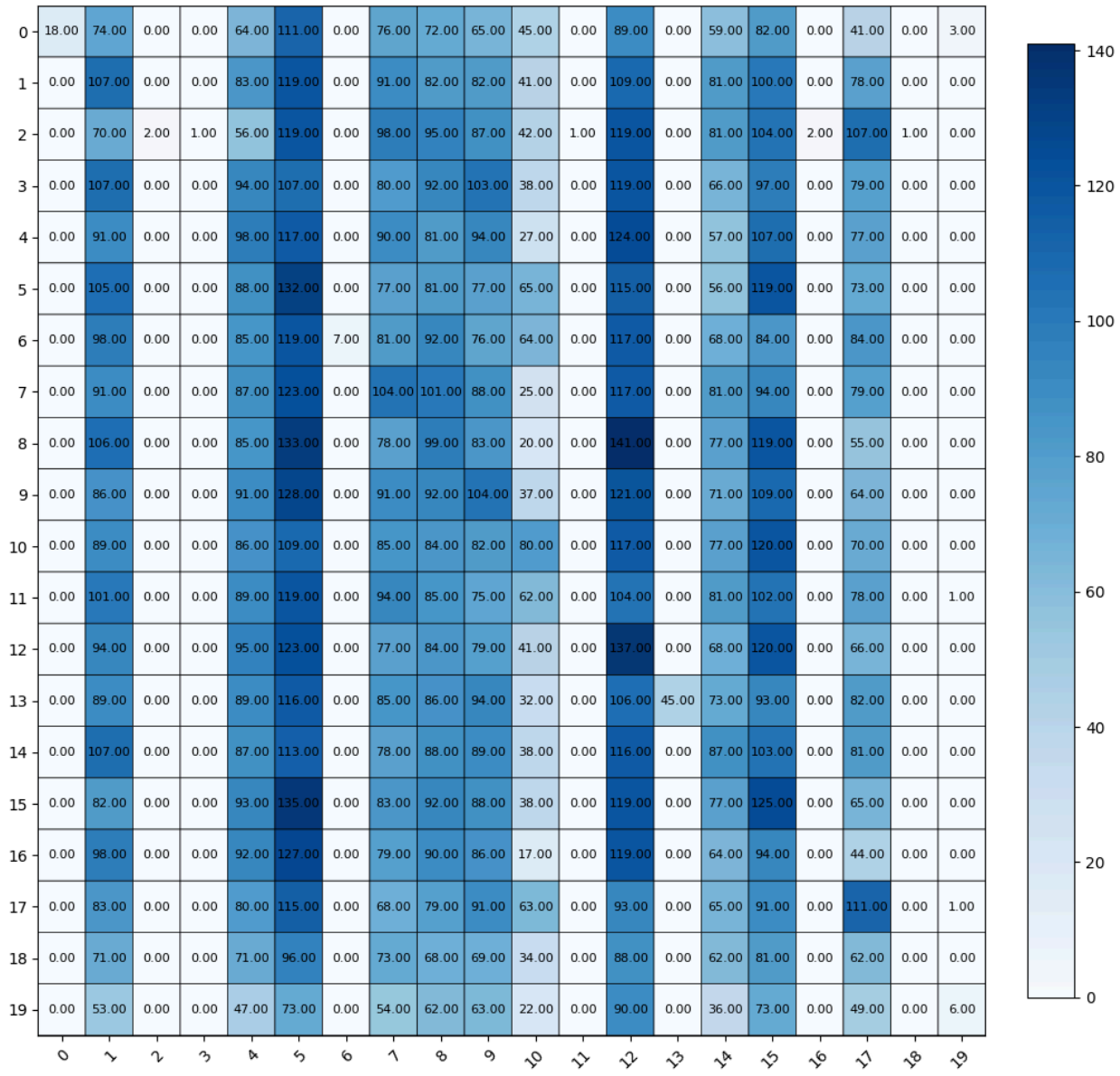


Figure 18: Permuted Contingency Matrix After Reducing Dimensionality with UMAP

QUESTION 12. Analyze the contingency matrices. Which setting works best and why? What about for each metric choice?

The Permuted Contingency Matrix shows a reasonable amount of high values along the diagonal, suggesting that there's some alignment between the true labels and the clusters.

With the cosine metric, the homogeneity, completeness, V-measure, and Adjusted Rand Index are significantly higher than when using the euclidean metric, suggesting that the cosine metric is more suitable for clustering this text data. It captures the similarity between documents based on the direction of their TF-IDF vectors rather than their magnitude. The performance does not drastically change with the number of components for the cosine metric, as seen in the relatively stable values for homogeneity, completeness, V-measure, and Adjusted Rand Index. This suggests that even with a smaller number of dimensions (5 or 20), UMAP is able to retain the essential structure necessary for effective clustering of this data. However the performance is consistently lower across all components.

n_components = 5 and *metric* = cosine yields the best clustering performance. This setting results in the highest values for all evaluation metrics except for Silhouette Coefficient, which is still relatively high. It indicates that a lower-dimensional space is sufficient to capture the structure and distribution of data for clustering purposes when using UMAP with the cosine metric. For this particular dataset, capturing the orientation similarity of the high-dimensional TF-IDF vectors is more important than capturing their magnitude differences. The cosine metric's insensitivity to vector magnitude makes it particularly well-suited for text clustering. The euclidean metric performs poorly compared to the cosine metric across all numbers of components. This is expected because the euclidean distance can be influenced by the magnitude of the TF-IDF vectors, which does not necessarily represent the similarity between documents.

QUESTION 13. So far, we have attempted K-Means clustering with 4 different representation learning techniques (sparse TF-IDF representation, PCA-reduced, NMF-reduced, UMAP-reduced). 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.

TF-IDF Representation

This approach provides a high-dimensional sparse representation, resulting in the best high-dimensional technique. The metrics indicate relatively balanced performance across homogeneity, completeness, and V-measure. ARI is quite high, suggesting that the clusters found have a reasonable correspondence with the true labels. However, the Silhouette Coefficient is very low, which means the clusters are not well-separated and could be of varying densities.

PCA-Reduced

PCA aims to retain the maximum variance in the data with fewer dimensions. The metrics for PCA-reduced data show a drop in performance across all metrics compared to the original TF-IDF representation, suggesting that the variance captured by PCA might not align well with the underlying cluster structure of the data.

NMF-Reduced

NMF forces a non-negative representation, which can be beneficial for interpretability. However, the clustering metrics for NMF-reduced data show poor performance across the board, with very low homogeneity, completeness, and ARI, indicating that the clusters generated do not correspond well to the true labels.

UMAP-Reduced

UMAP reduces dimensionality while preserving local and global structures and can use various distance metrics. Using cosine distance with UMAP yielded the best homogeneity, completeness, V-measure, and ARI out of all the methods tested. The Silhouette Coefficient is also substantially higher, likely due to better-defined clusters.

UMAP-reduced data with cosine distance stands out as the best approach for K-Means clustering on the 20-class text data. It has the highest homogeneity, completeness, V-measure, and ARI, indicating a strong correspondence between clusters and true labels, and the clusters are well-separated. PCA and NMF reductions are the techniques that performed weaker than the others, signified in their low performance metrics. The UMAP-reduced approach with cosine distance should be the preferred choice for K-Means clustering on this dataset..

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.

Ward Linkage performs significantly better than Single Linkage across all metrics. It produces clusters that are more homogenous, complete, and better matched to the true labels, as evidenced by the higher V-measure and ARI. The much higher Silhouette Coefficient signifies clusters are more compact and better separated using Ward Linkage. Single Linkage performs poorly, with very low scores in all metrics except for completeness, which is slightly higher due to the chaining effect that tends to merge clusters prematurely, leading to a few large clusters that encompass a range of classes. For agglomerative clustering with n_clusters=20 on the 20-class text data reduced with UMAP, Ward linkage is the superior choice. It creates more meaningful clusters that better correspond to the actual classes in the dataset, while Single Linkage produces clusters that do not capture the inherent structure of the data well and are negatively influenced by the chaining effect.

Homogeneity	0.4307792310838353
Completeness	0.46227934953742555
V-measure	0.4459737514668628
Adjusted Rand Index	0.28620133162894884
Silhouette Coefficient	0.34760433435440063

Table 10: Metrics for Ward Linkage

Homogeneity	0.009609925217071026
Completeness	0.12716210003525844
V-measure	0.017869418392102453
Adjusted Rand Index	0.0001243338715371397
Silhouette Coefficient	-0.37408116459846497

Table 11: Metrics for Single Linkage:

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.

Homogeneity	0.33405726746696973
Completeness	0.37634711456015657
V-measure	0.3539434493641508
Adjusted Rand Index	0.04881922256864343
Silhouette Coefficient	-0.04922362044453621

Table 12: Results for *min_cluster_size* = 20

Homogeneity	0.0006602952494961827
Completeness	0.014803402031630292
V-measure	0.0012642016796070616
Adjusted Rand Index	0.0000295610467019351
Silhouette Coefficient	0.8050383925437927

Table 13: Results for *min_cluster_size* = 100

Homogeneity	0.0006602952494961827
Completeness	0.014803402031630292
V-measure	0.0012642016796070616
Adjusted Rand Index	0.0000295610467019351
Silhouette Coefficient	0.8050383925437927

Table 14: Results for *min_cluster_size* = 200

Best model used: *min_cluster_size* = 20 with V-measure: 0.3539434493641508

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.

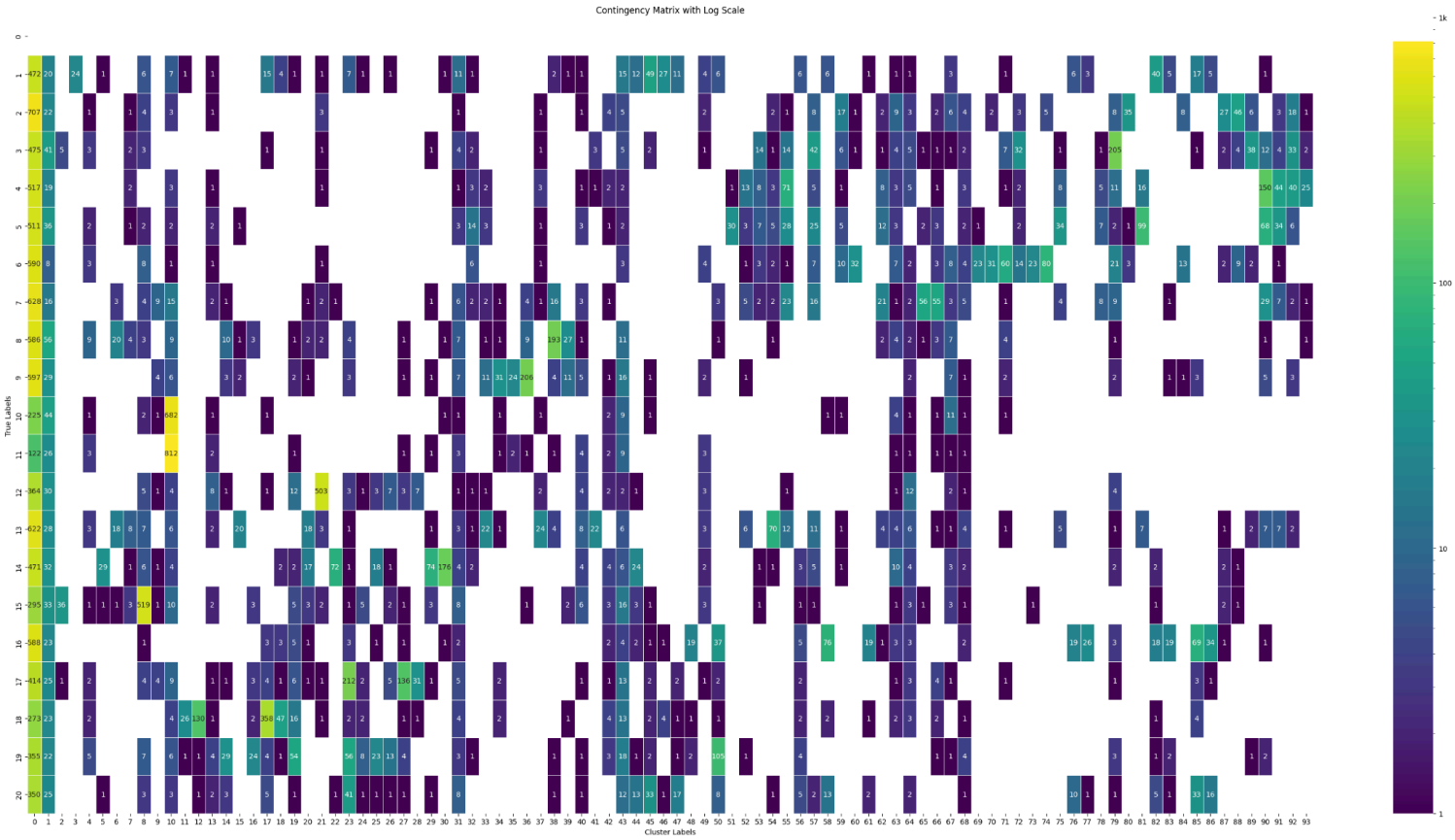


Figure 19: Contingency Matrix for *min_cluster_size* = 20 with V-measure: 0.354

As a result of running this line of code:

```
print(len(set(best_clusters)) - (1 if -1 in best_clusters else 0))
```

There are 93 clusters given by the model.

In HDBSCAN, a label of -1 indicates noise points. These are data points that the algorithm could not assign to any of the clusters with sufficient confidence, typically outliers or points that do not conform to the density-based clustering structure that HDBSCAN is looking for.

Each row in the contingency matrix corresponds to a true label from newsgroups_all.target. Each column corresponds to a cluster label assigned by the clustering model. The values in the matrix indicate the number of points from the dataset that belong to both the corresponding true label and cluster label.

In the generated contingency matrix, there are a large number of points concentrated in columns 1, 8 and 10, indicating that the cluster aligns well with a true class. The dispersed distribution suggests that the points of the true classes are spread out over multiple clusters. The -1 column shows how many points from each true label were considered noise by HDBSCAN. There is a high number of points in the -1 column, indicating that the model is either too conservative in forming clusters or that the data has many outliers or is not well-suited for density-based clustering. There is not a noticeable pattern of high values along the diagonal and low values off the diagonal as well as the heavily populated noise column both suggest that the HDBSCAN clustering method did not produce a strong clustering result.

QUESTION 17. Based on your experiments, which dimensionality reduction technique and clustering methods worked best together for 20-class text data and why?

For the 20-class text data, the UMAP reduction with cosine distance followed by K-Means clustering provided the best performance across these metrics, making it the preferred approach among the ones tested. Unlike the other methods, UMAP reduction Maximizes homogeneity, completeness, and V-measure, has a high ARI and a high Silhouette Coefficient as shown in Table 4. Together, these mean that the clusters generated are well-separated and internally cohesive and reliably correspond to the true classes. UMAP was effective in capturing the structure of the high-dimensional text data in a lower-dimensional space that aligns well with the assumptions of K-Means clustering.

QUESTION 18. Extra credit: If you can find creative ways to further enhance the clustering performance, report your method and the results you obtain.

The "Stabilized Ensemble Clustering on Dimensionality-Reduced Data" method is an ensemble-based approach that integrates the stability of multiple K-Means cluster assignments with the dimensionality reduction capabilities of PCA and UMAP. The process involves four main steps:

1. UMAP for Initial Reduction
    a. The high-dimensional data is initially reduced using UMAP with cosine similarity to capture the geometric structure of the data in a lower-dimensional space.

2. Normalization and PCA Processing
    a. The UMAP-reduced data is normalized to ensure each feature has equal weight, followed by PCA to condense the information into 10 principal components. The addition of this step refines the data representation, focusing on the most variance-explaining features which might be more relevant for clustering.

3. Ensemble Clustering with K-Means
    a. Multiple K-Means models are trained with different random initializations to capture a variety of potential cluster assignments. These models are independently fitted to the PCA-processed data.

4. Majority Voting on Cluster Assignments
    a. The cluster assignments from each K-Means model are aggregated, and the mode of these assignments is taken to determine the final cluster labels for each data point. This is to mitigate the randomness of K-Means initializations, hopefully making the clustering outcome more robust.

Adjusted Rand Index	0.23886243737162485
Normalized Mutual Information	0.4826972555778645
Silhouette Score	0.0789744434499523

Table 14: Output of Stabilized Ensemble Clustering on Dimensionality-Reduced Data Method on Full 20-category 20newsgroups dataset

The output metrics show that this method has achieved a moderate ARI, indicating a fair agreement between the clustering assignments and the ground truth. The NMI is relatively high, suggesting a good correlation between the clustering and the true labels, capturing the mutual information effectively. The Silhouette Score, while not high, indicates some level of cohesion and separation within the clusters.

QUESTION 19. In a brief paragraph discuss: 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, like other deep neural networks trained on large and diverse datasets such as ImageNet, learns a hierarchy of features through its layers. In the initial and intermediate layers, the network detects simple patterns like edges and textures that are not necessarily specific to the classes found in the training dataset, but in deeper layers it detects more complex features that represent higher-level abstractions. The discriminative power of features derived from a network trained on a broad dataset like ImageNet comes from this hierarchy of learned features. Even if the target classes in the pre-trained network differ from those in the custom dataset, the lower and mid-level features can still be relevant and useful for new image classification tasks. This concept is known as transfer learning, where knowledge gained while solving one problem is leveraged to solve a different related problem. Pre-trained models on ImageNet have been successfully applied to a wide range of image recognition tasks beyond the original 1000 classes of the dataset. For a custom dataset like the `tf_flowers` dataset, while the specific types of flowers may not have been in the ImageNet classes, the visual patterns that define what a flower looks like are likely to be similar to patterns the VGG network has learned. Thus, the network can effectively act as a feature extractor, providing a rich set of features that can be used to train a classifier or for clustering tasks on the custom dataset.

QUESTION 20. In a brief paragraph explain how the helper code base is performing feature extraction.

The helper code base performs feature extraction using a pre-trained VGG-16 CNN: It sets up data transformations for the images, loads the dataset, and creates a data loader. During feature extraction, it iterates through the dataset in batches, passes the images through the VGG-16 model to extract features, and stores these features along with their corresponding labels in memory. Finally, it saves the extracted features and labels for future use.

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 pixel height and pixel weight of each original image varies largely from around 240*160 to around 375*500; thus the images have roughly 40,000 to 190,000 pixels.

The VGG network extracts **4096 features per image**.

QUESTION 22. Are the extracted features dense or sparse? (Compare with sparse TF-IDF features in text.)

The extracted features from VGG are **dense** compared to sparse TF-IDF features in text.

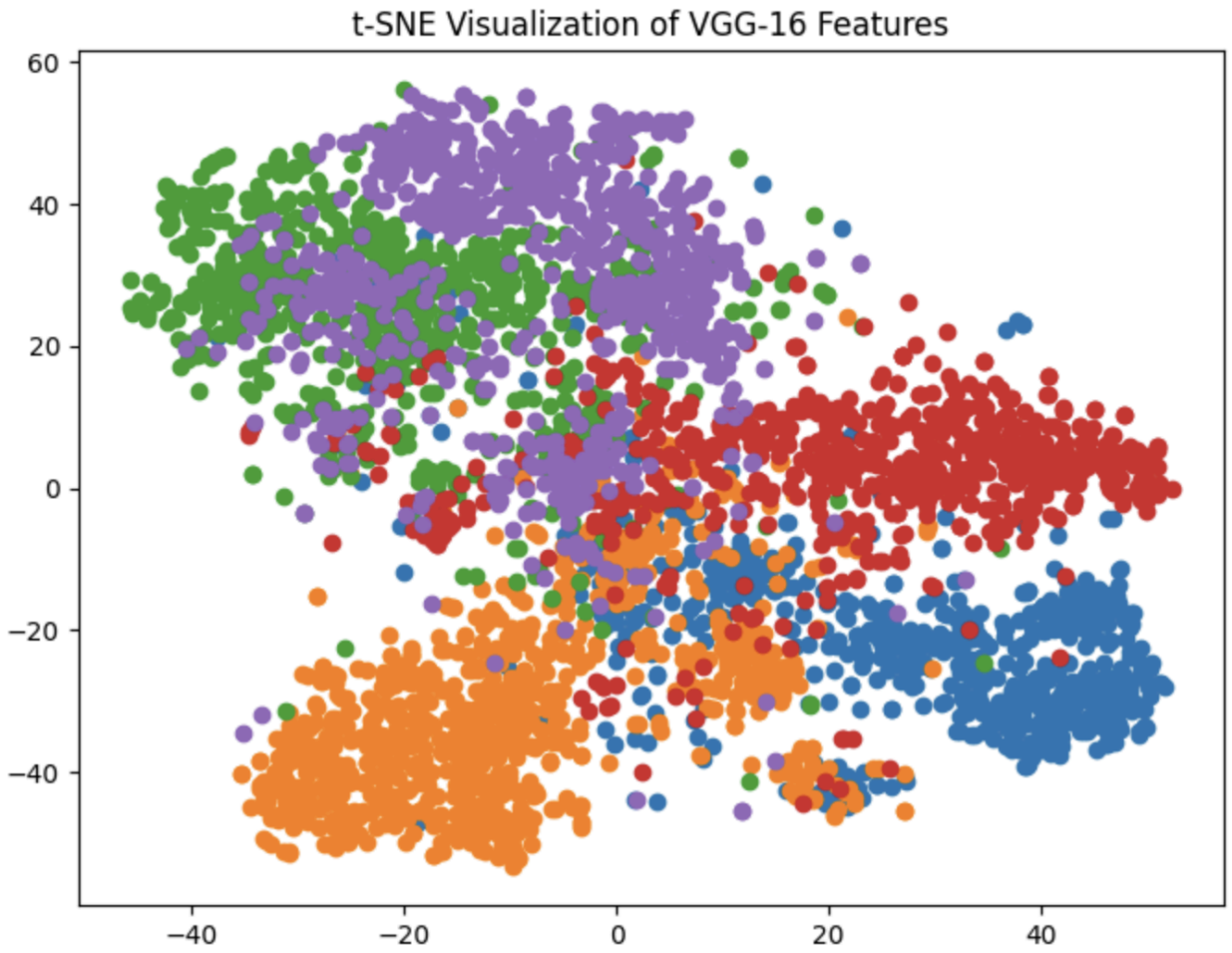
In TF-IDF, since not all terms are present in each document, the representation is typically sparse, meaning that most elements of the feature vector are zero due to the absence of many terms in each document.

While in VGG, each feature in the output feature vectors corresponds to a learned representation of some aspect or pattern detected in the input image, so the features are dense because most elements of the feature vector are non-zero and contain meaningful information about the image, such as edges, textures, and object parts.

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.

The following image is the t-SNE visualization of features.

We can observe that the five categories can be mainly separated, but in the middle there is a lot of overlapping, which means that in a 2-dimensional feature extraction, it’s hard to completely tell all the categories apart.



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.

```
Experiment (None, KMeans) Rand Score: 0.18919803381799868
Experiment (SVD, KMeans) Rand Score: 0.19123637967513135
Experiment (UMAP, KMeans) Rand Score: 0.40811041157693556
Experiment (Autoencoder, KMeans) Rand Score: 0.221027151179688

Experiment (None, Agglomerative) Rand Score: 0.24051166033092847
Experiment (SVD, Agglomerative) Rand Score: 0.2375609642467186
Experiment (UMAP, Agglomerative) Rand Score: 0.37604127761093387
Experiment (Autoencoder, Agglomerative) Rand Score: 0.271126906049534

Experiment (None, HDBSCAN)
- Best Rand Score (HDBSCAN): 0.029935533978548482
- Best min_cluster_size (HDBSCAN): 5
- Best min_samples (HDBSCAN): 2
```

```
Experiment (SVD, HDBSCAN)
- Best Rand Score (HDBSCAN): 0.025581182239683267
- Best min_cluster_size (HDBSCAN): 5
- Best min_samples (HDBSCAN): 3
Experiment (UMAP, HDBSCAN)
- Best Rand Score (HDBSCAN): 0.05435887929051603
- Best min_cluster_size (HDBSCAN): 10
- Best min_samples (HDBSCAN): 3
Experiment (Autoencoder, HDBSCAN)
- Best Rand Score (HDBSCAN): 0.024432358356232822
- Best min_cluster_size (HDBSCAN): 5
- Best min_samples (HDBSCAN): 2
```

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.

With the original VGG features, the test accuracy of MLP is 0.901.
When using the reduced-dimension features of PCA (n_components=100), the test accuracy of MLP is 0.910, roughly the same as the result on the original features.
However, when using the reduced-dimension features of PCA (n_components=2), the test accuracy of MLP is 0.530, which **suffers significantly from** the reduced dimension representation.
Therefore, we can notice that if the dimension is reduced less significantly (e.g. 100 components), the result will not suffer a lot. However, if the dimension is reduced largely (e.g. reduced to 2 components), the result will suffer significantly. This observation also aligns with our results in Question 24. In Question 24, we can notice that some results are about the same as the results from the original feature, while other dimensional reducing techniques may lead to a worse result.

QUESTION 26. Try to construct various text queries regarding types of Pokemon (such as ”type: Bug”, ”electric type Pok´emon” or ”Pok´emon 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 Pokemon for type Bug, Fire and Grass. For each of the constructed query, please plot the five most relevant Pokemon horizontally in one figure with following specifications:

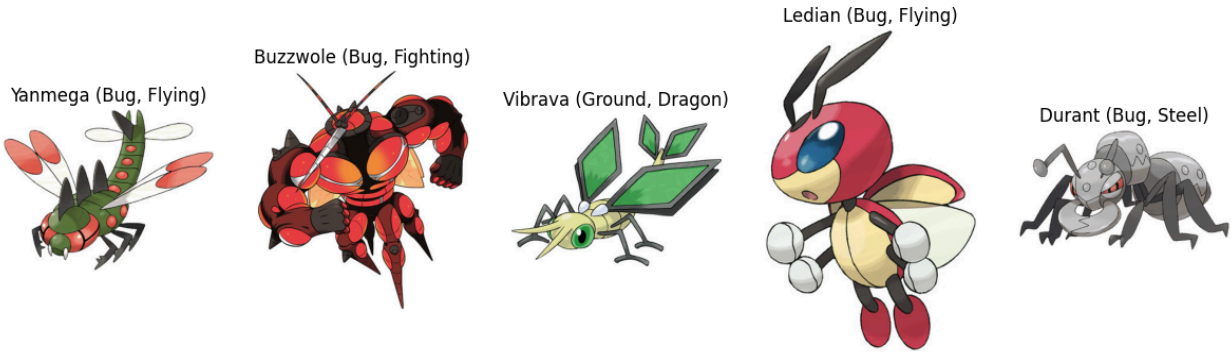
- the title of the figure should be the query you used;
- the title of each Pokemon should be the name of the Pokemon and its first and second type.

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

After multiple attempts, we decided to use the query “type: {type_name}” as the most suitable query, and achieved the following results.

We can notice that most of the images have the correct type regarding our query, while the few ones that does not correspond to the query also demonstrate some reasonable attributes with respect to the query type (e.g. *Vibrava* looks like a kind of *bug*, and *Corviknight* has an overall *dark* color). Therefore, the model can be conceived as mostly correct.

type: Bug



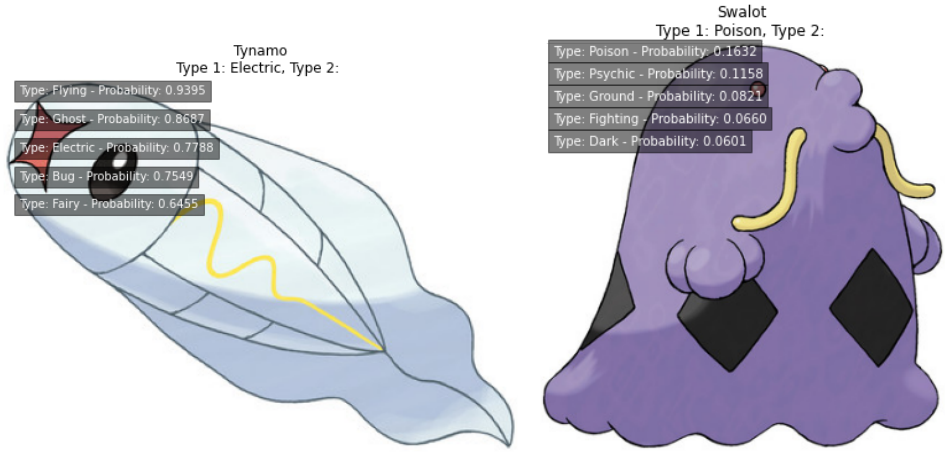
type: Fire

type: Grass

QUESTION 27. Randomly select 10 Pokemon images from the dataset and use CLIP to find the most relevant types (use your preferred template, e.g ”type: Bug”). For each selected Pokemon, please plot it and indicate:

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

Shown below are the 10 images with their five most relevant types. Most images have the correct type predicted as the top 5, proving the accuracy of the model.




Slakoth
Type 1: Normal, Type 2:
Type: Normal - Probability: 0.0713
Type: Psychic - Probability: 0.0446
Type: Ground - Probability: 0.0439
Type: Dragon - Probability: 0.0432
Type: Grass - Probability: 0.0388
Tirtouga
Type 1: Water, Type 2: Rock
Type: Water - Probability: 0.5972
Type: Fighting - Probability: 0.1418
Type: Steel - Probability: 0.1271
Type: Normal - Probability: 0.0916
Type: Electric - Probability: 0.0874
Purrloin
Type 1: Dark, Type 2:
Type: Psychic - Probability: 0.2302
Type: Dark - Probability: 0.1487
Type: Poison - Probability: 0.1332
Type: Fire - Probability: 0.0916
Type: Dragon - Probability: 0.0821
Gigalith
Type 1: Rock, Type 2:
Type: Rock - Probability: 1.0000
Type: Steel - Probability: 0.7549
Type: Ground - Probability: 0.7432
Type: Ice - Probability: 0.6768
Type: Dark - Probability: 0.5435
Ivysaur
Type 1: Grass, Type 2: Poison
Type: Grass - Probability: 0.3037
Type: Poison - Probability: 0.5698
Type: Fighting - Probability: 0.4233
Type: Fairy - Probability: 0.2451
Type: Water - Probability: 0.2031
Dubwool
Type 1: Normal, Type 2:
Type: Fire - Probability: 0.1088
Type: Dark - Probability: 0.1071
Type: Grass - Probability: 0.0990
Type: Normal - Probability: 0.0783
Type: Dragon - Probability: 0.0573
Surskit
Type 1: Bug, Type 2: Water
Type: Normal - Probability: 0.1354
Type: Electric - Probability: 0.0629
Type: Psychic - Probability: 0.0538
Type: Bug - Probability: 0.0530
Type: Ice - Probability: 0.0239
Treecko
Type 1: Grass, Type 2:
Type: Dragon - Probability: 0.1710
Type: Psychic - Probability: 0.0506
Type: Normal - Probability: 0.0426
Type: Fighting - Probability: 0.0413
Type: Fire - Probability: 0.0348

QUESTION 28. In the first and second question, we investigated how CLIP creates 'clusters' by mapping images and texts of various Pokemon into a high-dimensional space and explored neighborhood of these items in this space. For this question, please use t-SNE to visualize image clusters, specifically for Pokemon 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 Pokemon'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 Pokemon types make sense to you.

From the plot, we cannot observe very discernible clusters among the Bug, Fire, and Grass type Pokemon, which means that there might not be a clear cut among the three types, and many pokemons may have either two of the three types. We can also notice that the Grass type and the Bug type appears to be more intermingled rather than with the Fire type, which means that pokemons are more likely to obtain the Grass type and the Bug type at the same time (e.g. a green bug).