Project 2: Data Representations and Clustering

Lauren Mizner (#005225768)

Part 1: Clustering on Text Data

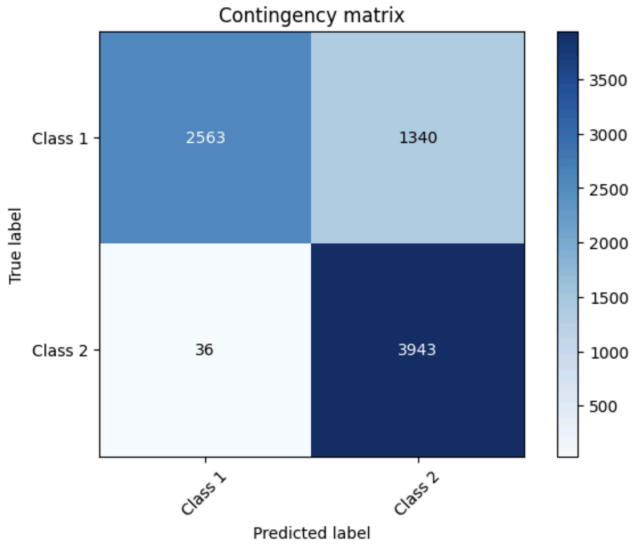
Question 1

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

(7882, 18469)

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?



The contingency matrix does not have to be square-shaped. In fact, it is typically rectangular-shaped, with rows to represent the true class labels and columns to represent the cluster labels. Each cell in the matrix indicates the total count of samples that belong to both the true class label and the cluster label.

Question 3

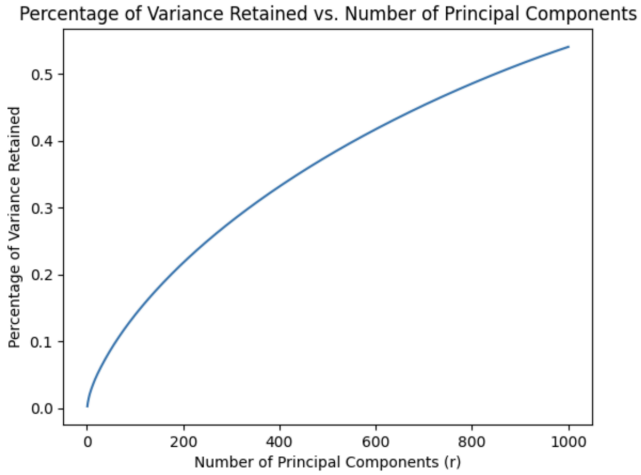
Report the 5 clustering measures explained in the introduction for K-means clustering.

Evaluation Metrics Scores:
Homogeneity Score = 0.4176
Completeness Score = 0.4565
V-Measure Score = 0.4362
Adjusted Rand Index (ARI) = 0.4235

Adjusted Mutual Information (AMI) Score = 0.4362

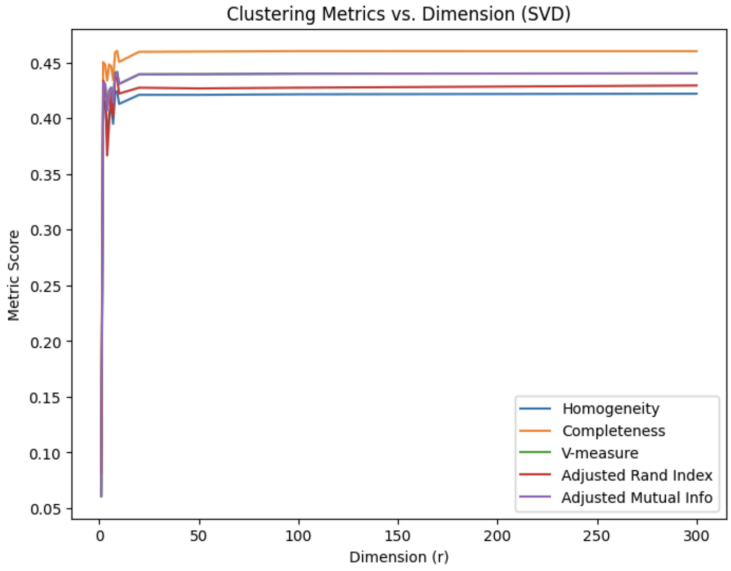
Question 4

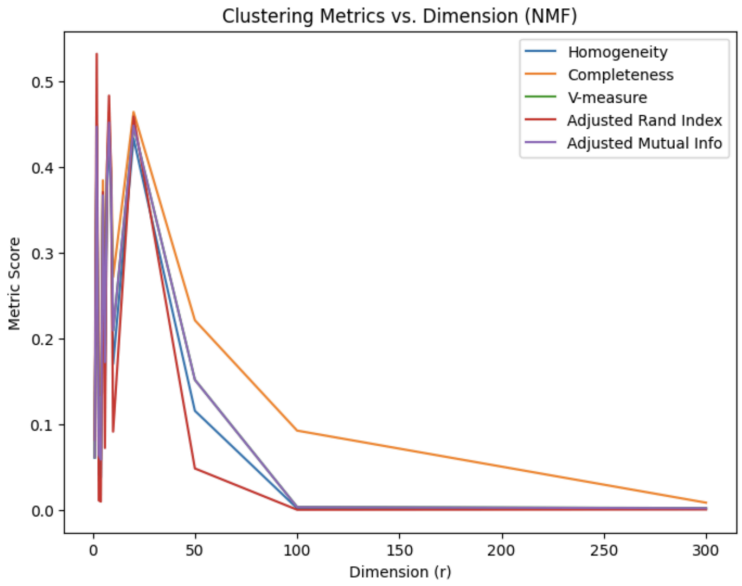
Report the plot of the percentage of variance that the top r principal components retain v.s. R, for r = 1 to 1000.



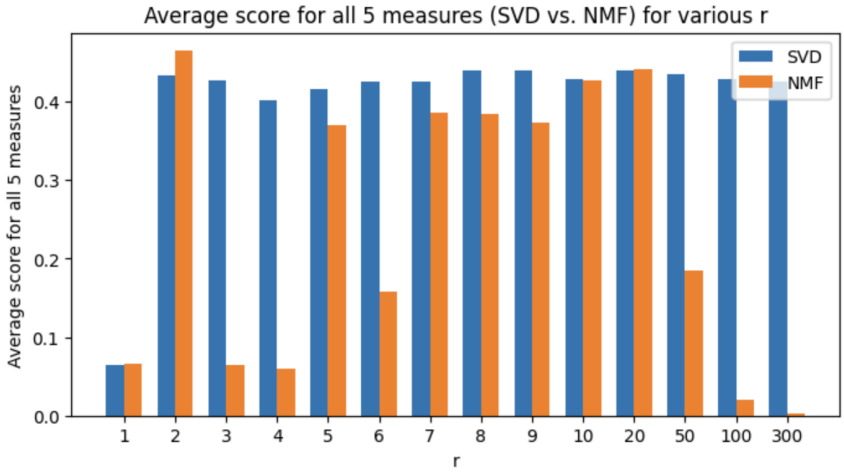
Question 5

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.





To determine a good choice of r, we averaged the 5 evaluation metrics for each r value to see which r value produces the best overall results. Based on the results obtained from the chart below, the best r value for SVD is r = 20 and the best r value for NMF is r = 2.



Question 6

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

The non-monotonic behavior of the clustering metrics can be attributed to properties such as overfitting and curse of dimensionality. In general, as we increase the number of dimensions, the model should capture more complex patterns in the data leading to better performance in clustering. However, at a certain point, adding more dimensions can cause the model to begin overfitting, where the model begins to capture noise or irrelevant patterns in the data. Another reason may be due to the curse of dimensionality, which says that as the dimensionality increases, the data becomes more sparse, leading to increased distance between data points and clusters, making it harder for the model to identify meaningful clusters.

Question 7

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

In general higher values indicate better performance for all of the evaluation metrics in question. The average scores across all r values for SVD and NMF are shown below. For all evaluation metrics, there was better performance achieved through the K-Means clustering method used in Question 3. On the other hand, if we compare the best evaluation metrics calculated for SVD with an r of 8 and NMF with an r of 2, the SVD and NMF K-Means clustering results achieve slightly better performance than that achieved through the K-Means clustering method used in Question 3.

Average evaluation metric scores for SVD:
Homogeneity Score = 0.3865
Completeness Score = 0.4228
V-Measure Score = 0.4038
Adjusted Rand Index (ARI) = 0.3941
Adjusted Mutual Information (AMI) Score = 0.4038

Average evaluation metric scores for NMF:
Homogeneity Score = 0.2262
Completeness Score = 0.2745
V-Measure Score = 0.2412
Adjusted Rand Index (ARI) = 0.2315
Adjusted Mutual Information (AMI) Score = 0.2411

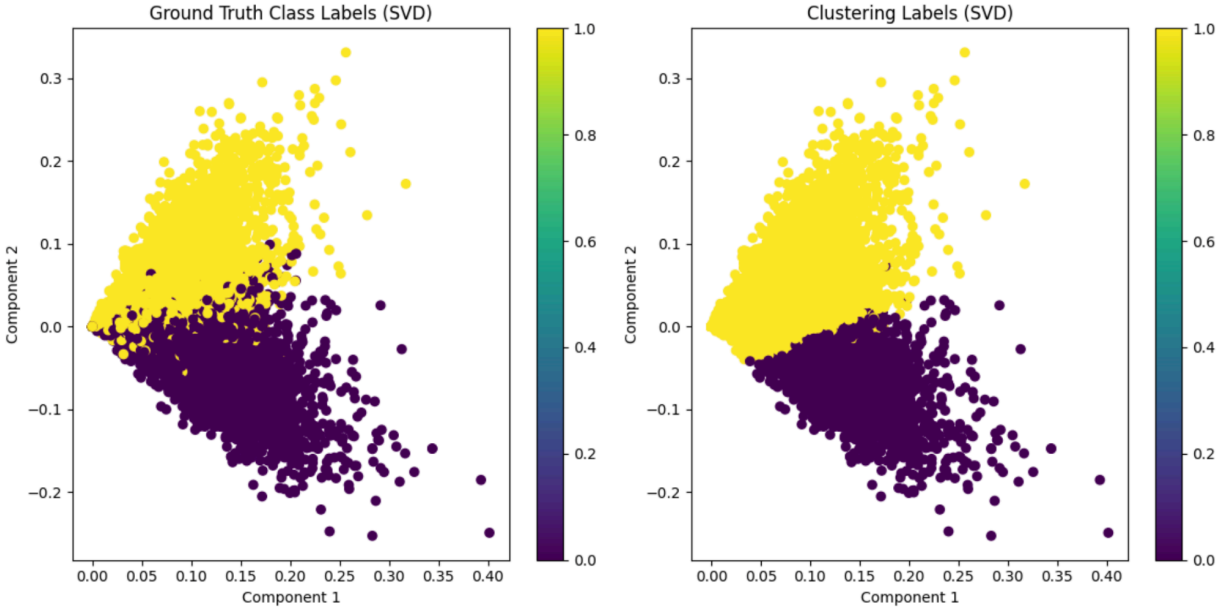
Best evaluation metric scores for SVD (at r = 20):
Homogeneity Score = 0.4234
Completeness Score = 0.4607
V-Measure Score = 0.4413
Adjusted Rand Index (ARI) = 0.4332
Adjusted Mutual Information (AMI) Score = 0.4412

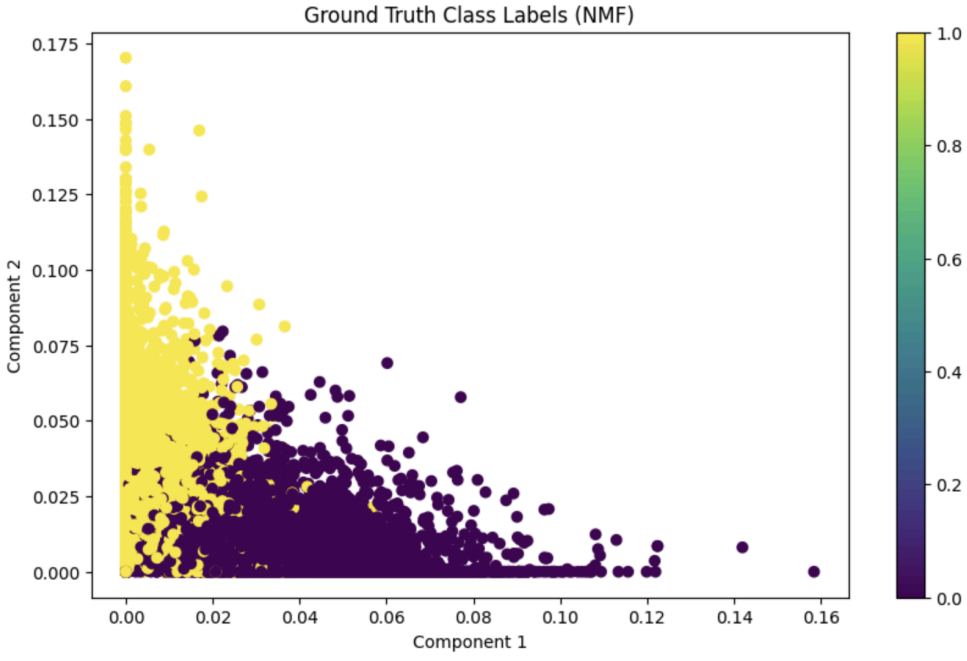
Best evaluation metric scores for NMF (at r = 2):
Homogeneity Score = 0.4451
Completeness Score = 0.4500
V-Measure Score = 0.4475
Adjusted Rand Index (ARI) = 0.5325
Adjusted Mutual Information (AMI) Score = 0.4475

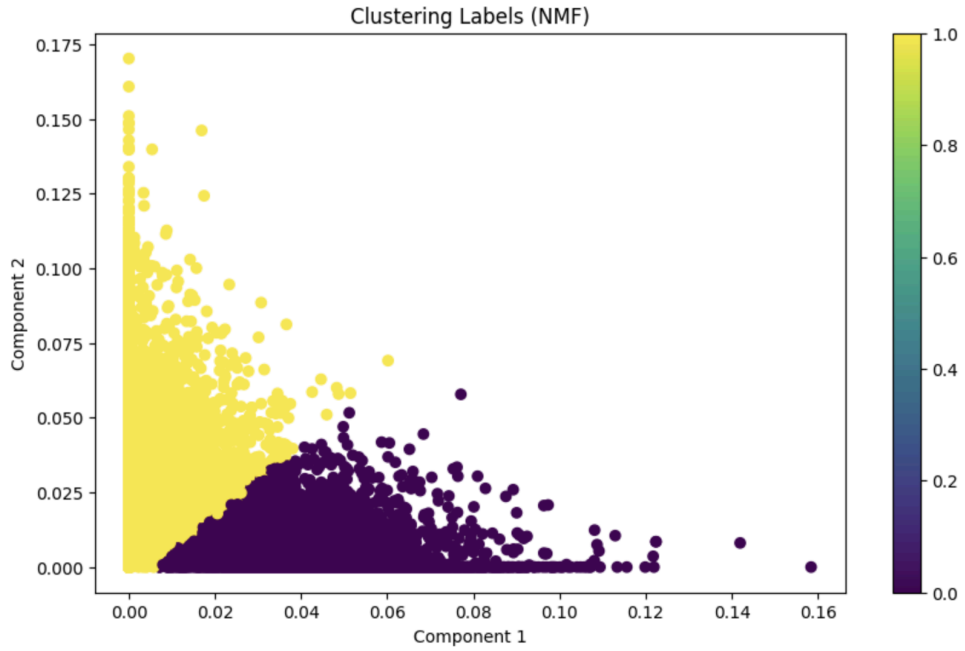
Question 8

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







Question 9

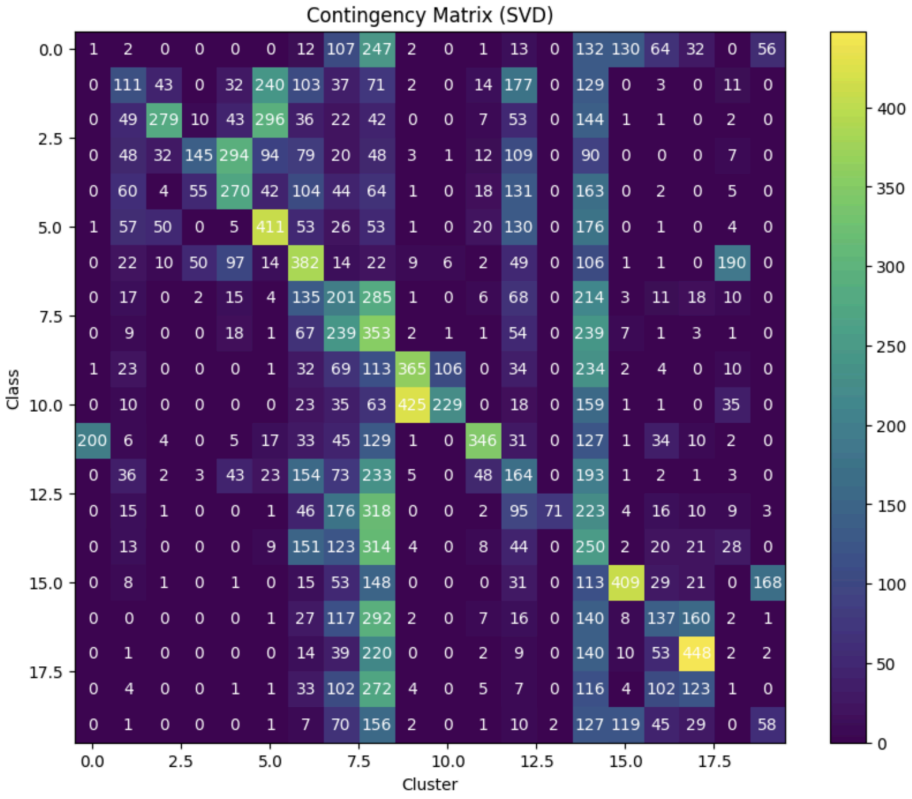
What do you observe in the visualization? How are the data points of the two classes distributed? Is distribution of the data ideal for K-Means clustering?

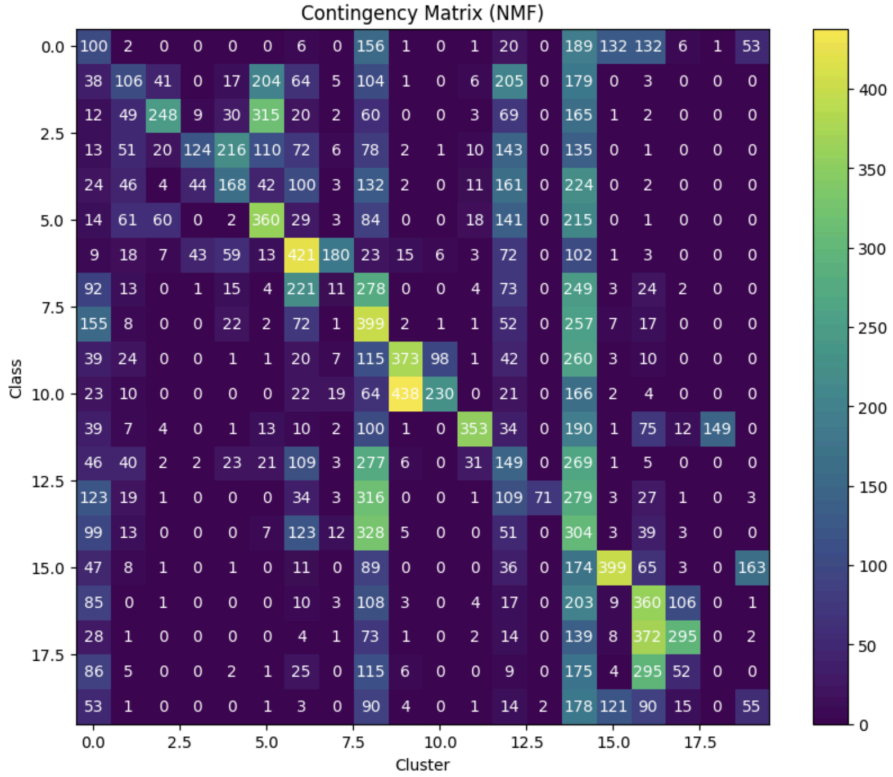
For SVD, the clustering is evenly split with one cluster forming above y = 0 and one cluster forming below y = 0, roughly. For NMF, the clustering is evenly split with one cluster forming above y = x and one cluster forming below y = x, roughly. For both methods, the clustering of the two classes is evenly distributed and well-separated, making it ideal for K-Means clustering.

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).

Evaluation Metric Scores for SVD:
Homogeneity Score = 0.2576
Completeness Score = 0.2931
V-Measure Score = 0.2742
Adjusted Rand Index (ARI) = 0.0822
Adjusted Mutual Information (AMI) Score = 0.2717


Contingency Matrix (SVD)
Class
Cluster
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0
50
100
150
200
250
300
350
400


Contingency Matrix (NMF)
Class
Cluster
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0
50
100
150
200
250
300
350
400

Evaluation Metric Scores for NMF:
Homogeneity Score = 0.2563
Completeness Score = 0.2974
V-Measure Score = 0.2753
Adjusted Rand Index (ARI) = 0.0780
Adjusted Mutual Information (AMI) Score = 0.2727

Question 11

Reduce the dimension of your dataset with UMAP. Consider the following settings: n_components = [5, 20, 200], metric = “cosine” vs “euclidean” (If “cosine” metric fails, please look at the FAQ at the end of the spec). Report the permuted contingency matrix and the five clustering evaluation metrics for the different combinations (6 combinations).

Evaluation Metric Scores UMAP (Cosine, 5):
Homogeneity Score = 0.4451
Completeness Score = 0.4605
V-Measure Score = 0.4526
Adjusted Rand Index (ARI) = 0.3048
Adjusted Mutual Information (AMI) Score = 0.4508

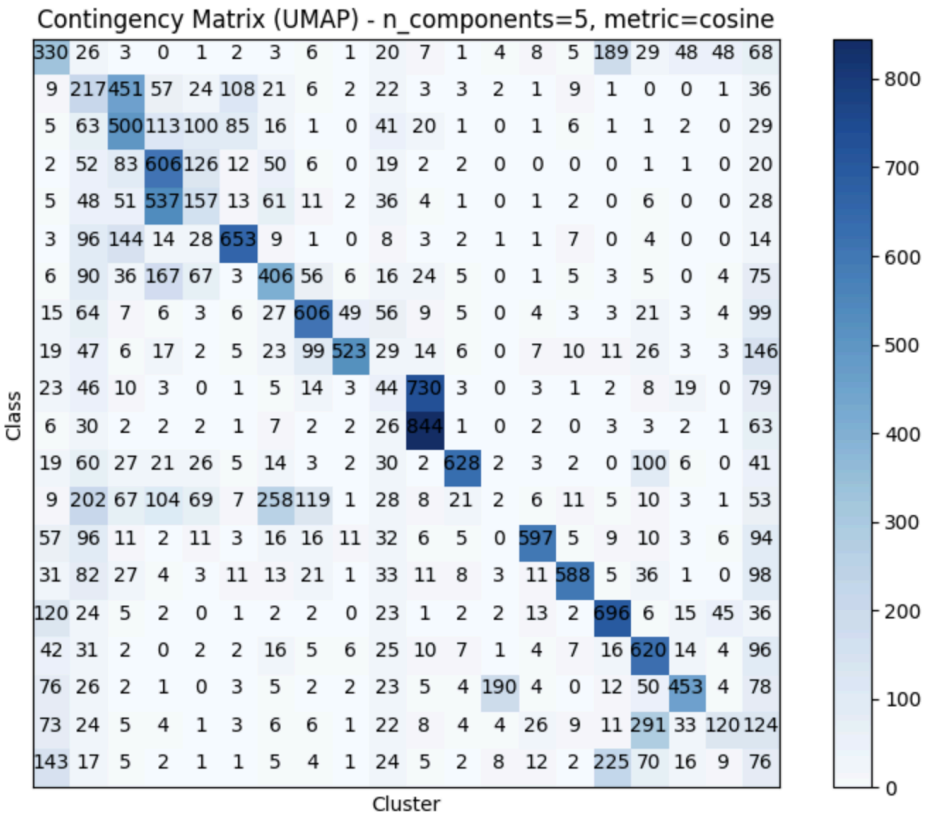
Evaluation Metric Scores UMAP (Cosine, 20):
Homogeneity Score = 0.4608
Completeness Score = 0.4708
V-Measure Score = 0.4657
Adjusted Rand Index (ARI) = 0.3330
Adjusted Mutual Information (AMI) Score = 0.4640

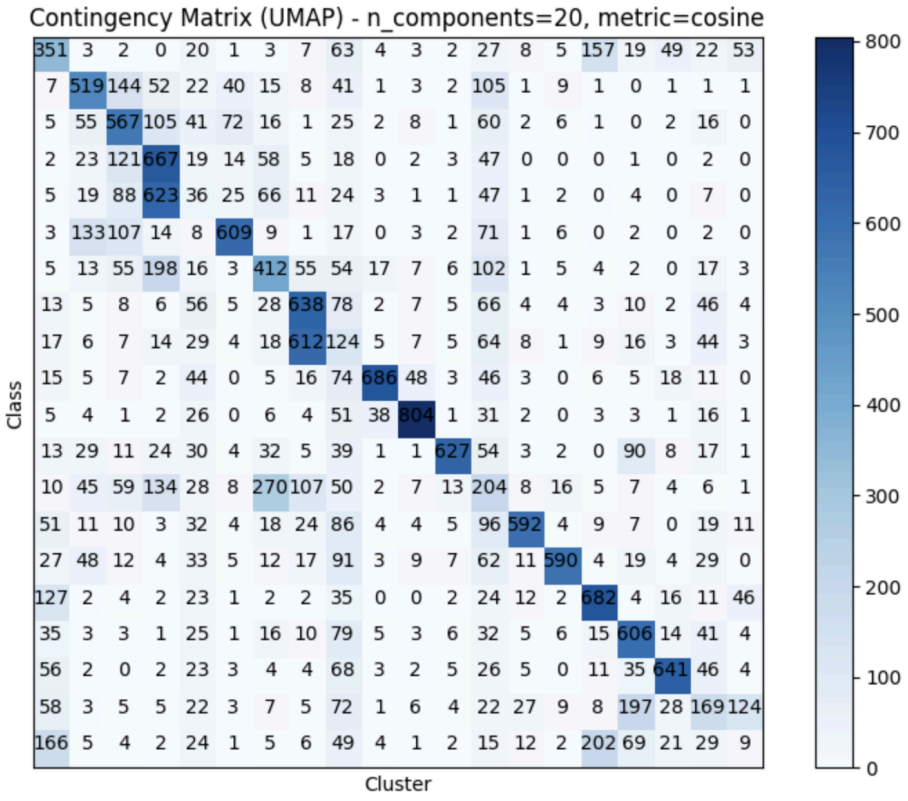
Evaluation Metric Scores UMAP (Cosine, 200):
Homogeneity Score = 0.4409
Completeness Score = 0.4606
V-Measure Score = 0.4505
Adjusted Rand Index (ARI) = 0.3039
Adjusted Mutual Information (AMI) Score = 0.4487

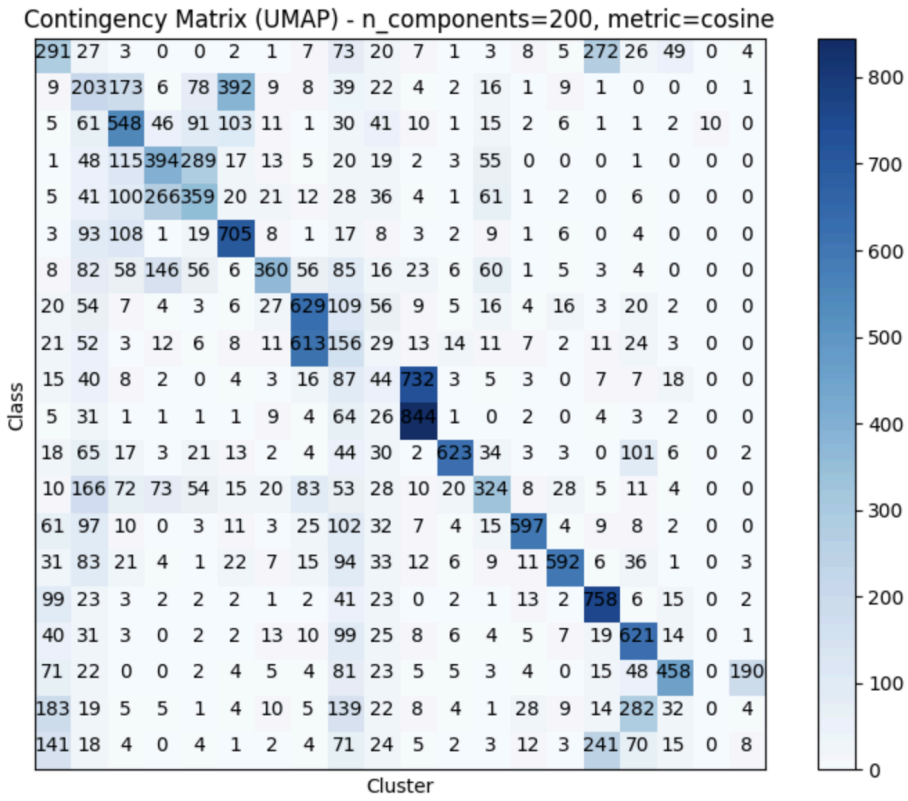
Evaluation Metric Scores UMAP (Euclidean, 5):
Homogeneity Score = 0.0096
Completeness Score = 0.0103
V-Measure Score = 0.0099
Adjusted Rand Index (ARI) = 0.0006
Adjusted Mutual Information (AMI) Score = 0.0066

Evaluation Metric Scores UMAP (Euclidean, 20):
Homogeneity Score = 0.0092
Completeness Score = 0.0102
V-Measure Score = 0.0097
Adjusted Rand Index (ARI) = 0.0007
Adjusted Mutual Information (AMI) Score = 0.0063

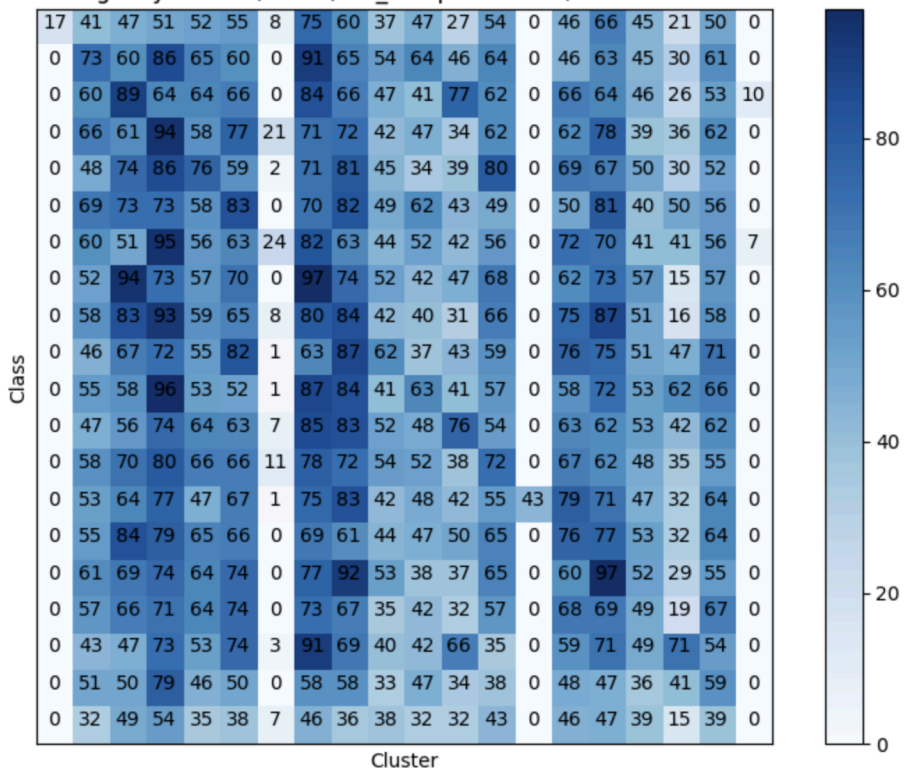
Evaluation Metric Scores UMAP (Euclidean, 200):
Homogeneity Score = 0.0097
Completeness Score = 0.0109
V-Measure Score = 0.0103
Adjusted Rand Index (ARI) = 0.0008
Adjusted Mutual Information (AMI) Score = 0.0069



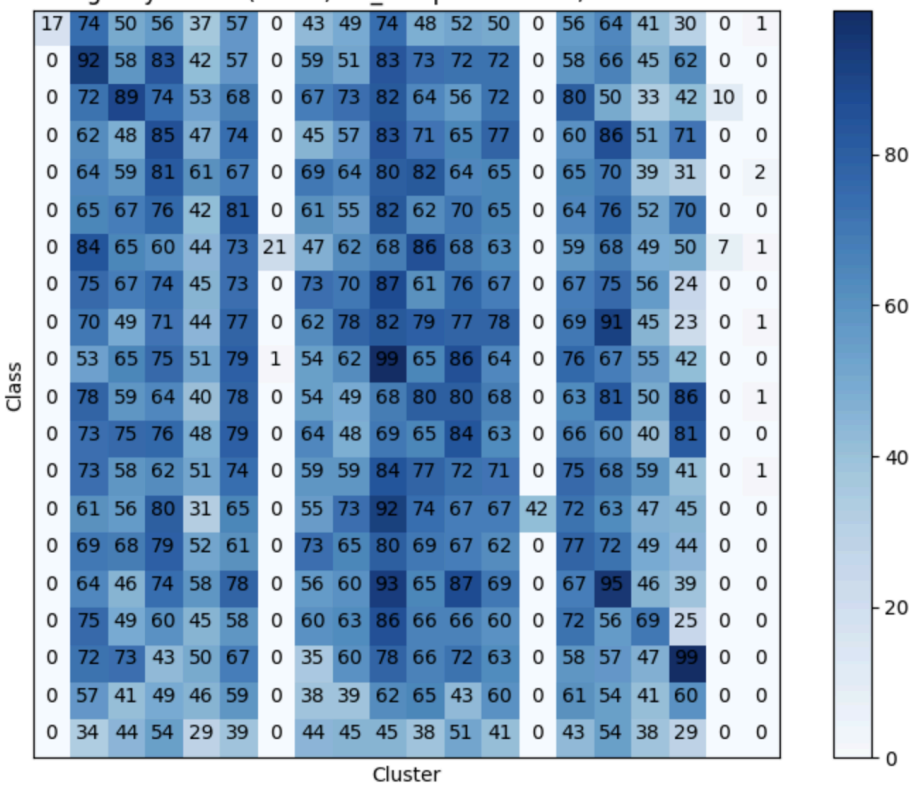

Contingency Matrix (UMAP) - n_components=20, metric=cosine
Class
Cluster
0
100
200
300
400
500
600
700
800

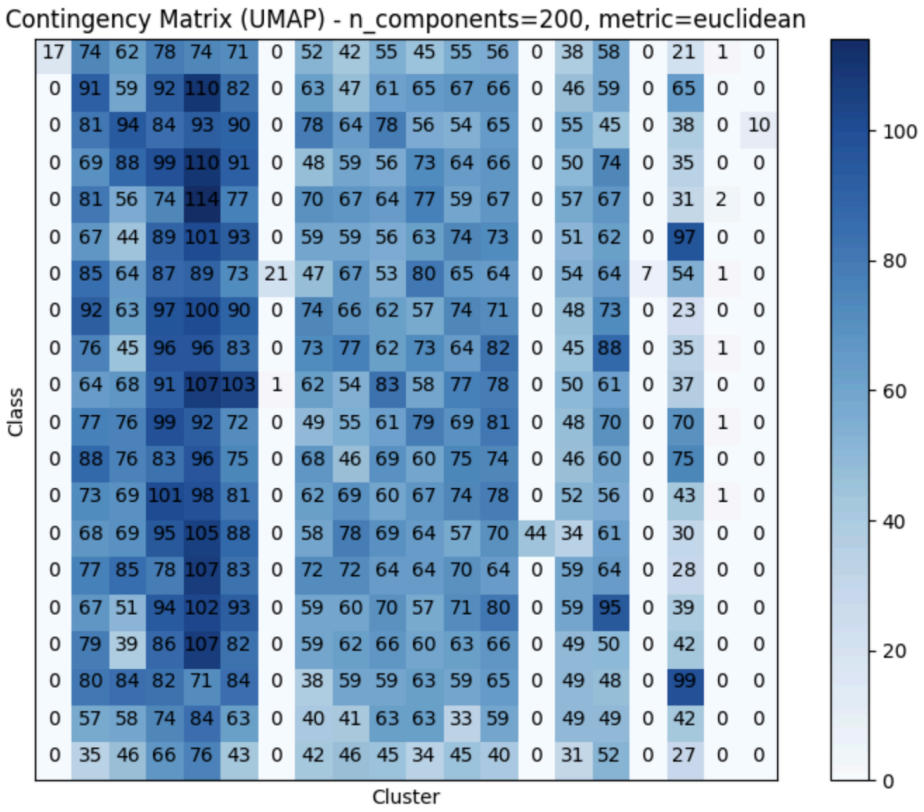

Contingency Matrix (UMAP) - n_components=200, metric=cosine
Class
Cluster
0
100
200
300
400
500
600
700
800

Contingency Matrix (UMAP) - n_components=5, metric=euclidean



Contingency Matrix (UMAP) - n_components=20, metric=euclidean





Question 12

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

From the contingency matrices, we can see that the cosine metric clearly outperforms the euclidean metric as we can see a strong diagonal forming within the matrices for the cosine metric, while no clear pattern is being developed among the matrices for the euclidean metric. From the evaluation metrics, it can be seen that the cosine metric with 20 components performs the best overall, while 200 components performs the best specifically for the euclidean metric.

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.

From the previous evaluations utilizing TF-IDF, SVD-reduced, NMF-reduced, and UMAP-reduced, the best overall results were seen with UMAP reduction, specifically when utilizing the cosine metric.

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.

From the results below, we can see that ward linkage out performs single linkage.

Evaluation Metric Scores for Ward Linkage:
Homogeneity Score = 0.4631
Completeness Score = 0.4690
V-Measure Score = 0.4661
Adjusted Rand Index (ARI) = 0.3452
Adjusted Mutual Information (AMI) Score = 0.4643

Evaluation Metric Scores Single Linkage:
Homogeneity Score = 0.0255
Completeness Score = 0.3628
V-Measure Score = 0.0476
Adjusted Rand Index (ARI) = 0.00079
Adjusted Mutual Information (AMI) Score = 0.0413

Question 15

Apply HDBSCAN on UMAP-transformed 20-category data. Use min_cluster_size = 100. Vary the min_cluster_size among 20, 100, and 200, and report your findings in terms of the five clustering evaluation metrics - you will plot the best contingency matrix in the next question.

Evaluation Metric Scores for Minimum Cluster Size = 20:
Homogeneity Score = 0.4318
Completeness Score = 0.3674
V-Measure Score = 0.3970
Adjusted Rand Index (ARI) = 0.0578
Adjusted Mutual Information (AMI) Score = 0.3783

Evaluation Metric Scores for Minimum Cluster Size = 100:
Homogeneity Score = 0.0131
Completeness Score = 0.4626
V-Measure Score = 0.0255
Adjusted Rand Index (ARI) = 0.00055

Adjusted Mutual Information (AMI) Score = 0.0249

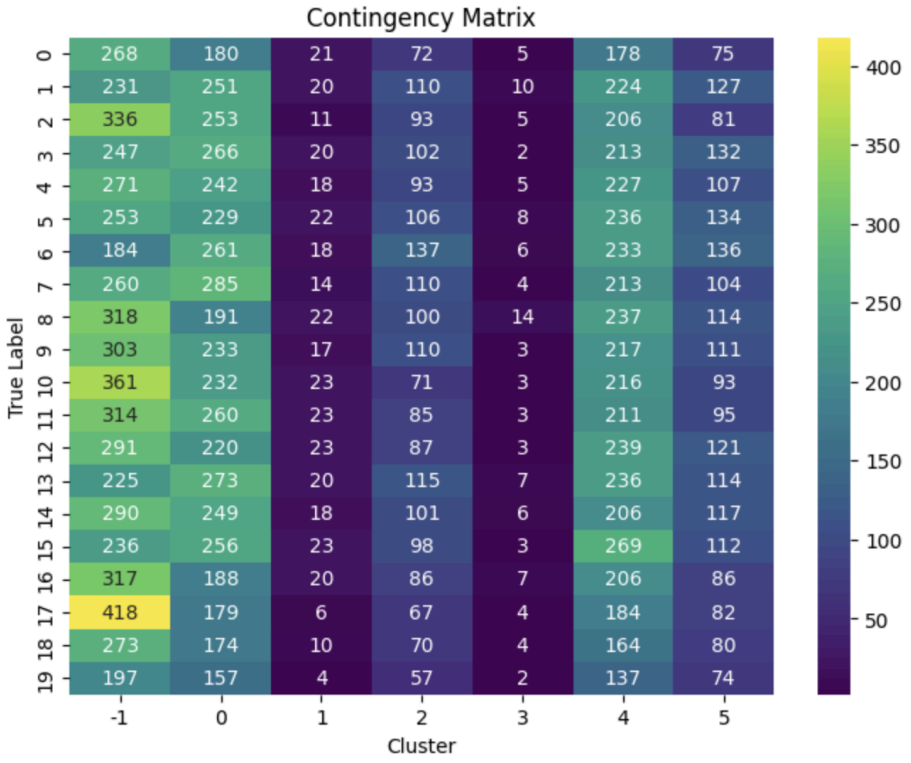
Evaluation Metric Scores for Minimum Cluster Size = 200:
Homogeneity Score = 0.0131
Completeness Score = 0.4626
V-Measure Score = 0.0255
Adjusted Rand Index (ARI) = 0.00055
Adjusted Mutual Information (AMI) Score = 0.0249

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.

Number of Clusters = 6

Some points may not fit well into any cluster and are considered to be outliers or noisy data. These data points are assigned to -1 label to show that they don't belong to any cluster.



Question 17

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

Based on the previous experiments, the best performance is seen with UMAP (cosine) dimensionality reduction and K-Means clustering and UMAP (cosine) dimensionality reduction and ward linkage Agglomerative clustering. In the results shown below, the two technique/method combinations produce very similar results with each performing slightly better than the other depending on the evaluation metric in question.

Evaluation Metric Scores for UMAP (cosine) & Ward Linkage Agglomerative Clustering:
Homogeneity Score = 0.4631
Completeness Score = 0.4690
V-Measure Score = 0.4661
Adjusted Rand Index (ARI) = 0.3452
Adjusted Mutual Information (AMI) Score = 0.4643

Evaluation Metric Scores UMAP (cosine) & K-Means Clustering:
Homogeneity Score = 0.4608
Completeness Score = 0.4708
V-Measure Score = 0.4657
Adjusted Rand Index (ARI) = 0.3330
Adjusted Mutual Information (AMI) Score = 0.4640

Question 18

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

One option could be utilizing an autoencoder similar to how we do in part 2 of this project.

Part 2: Deep Learning and Clustering of Image Data

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?

This is due to the concept of transfer learning, which utilizes the knowledge learned from the original dataset and applies it to a new dataset. The features learned by a VGG network, such as colors, textures, edges, etc., should be transferable knowledge that can be applied to a different dataset. So regardless of the specific classes used within the training data, it should be learning how to correctly distinguish between objects based on those features or characteristics.

Question 20

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

The dataset of images first go through resizing, cropping, and normalizing in preparation for the model. The dataset is then processed through a data loader to produce batches of images to be passed through the “FeatureExtractor” function. The “FeatureExtractor” function passes images through the VGG network’s convolutional/feature layers, pooling layers, and the first part of its fully connected layers in order to extract features and their corresponding labels, which are then saved to a file for further 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 original images are of varying sizes, but they are all resized to be a 224 x 224 or 50176 pixel image. From the VGG features, we can see that the sequential classifier produces a final fully connected layer with a 4096-dimensional feature vector, which can be utilized for feature extraction.

Question 22

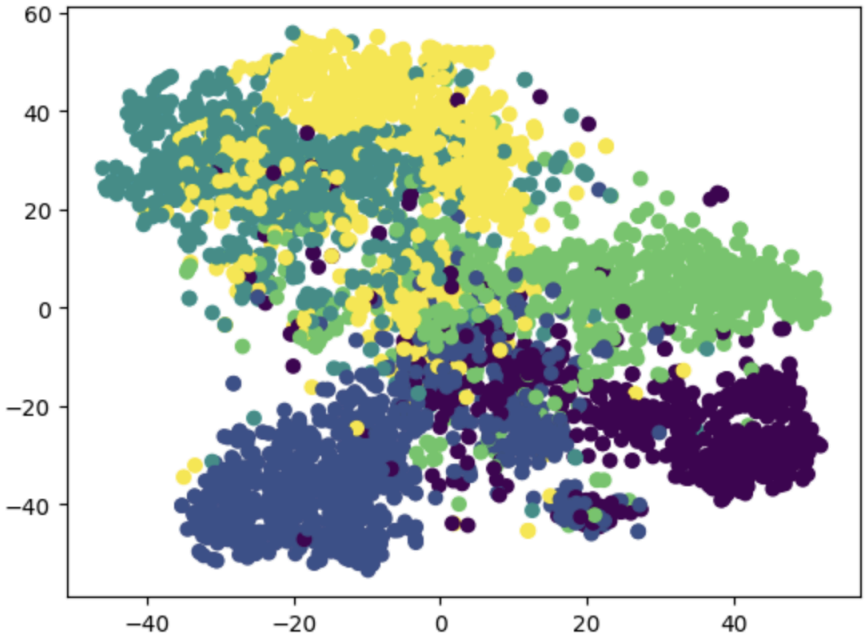
Are the extracted features dense or spares? (Compare with sparse TF-IDF features in text).

Typically, the extracted features from a convolutional neural network, such as VGG, are dense. The network architecture also indicates dense features as each layer in the architecture applies a convolutional layer followed by an activation function, in this case ReLU, and then a pooling operation which transforms the data into feature maps.

On the other hand, sparse TF-IDF features in text create matrices where each row corresponds to a document and each column corresponds to a term. This leads to a lot of zero entries amongst the matrix, making it sparse.

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 observation, we can see that there are 5 distinct clusters formed in the figure above. While most data points do fall within one of these distinct clusters, we do see quite a few points overlapping into different clusters showing that the clusters may not be as well separated as we'd like. This may be attributed to noisy data or it may indicate that another clustering method may be better suited for this data set.

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.

Dimensionality Reduction	Clustering	Rand Score
None	K-Means	0.193

None	Agglomerative Clustering	0.218
None	HDBSCAN	0.007
SVD	K-Means	0.190
SVD	Agglomerative Clustering	0.202
SVD	HDBSCAN	0.006
UMAP	K-Means	0.464
UMAP	Agglomerative Clustering	0.484
UMAP	HDBSCAN	0.095
Autoencoder	K-Means	0.230
Autoencoder	Agglomerative Clustering	0.250
Autoencoder	HDBSCAN	0.008

In terms of rand score, the best results for all clustering methods was seen when using the UMAP dimensionality reduction. The best overall combination of dimensionality reduction technique and clustering method was the combination of UMAP and Agglomerative clustering using the cosine metric.

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.

Accuracy of MLP Classifier on the original VGG features = 90.46%

Accuracy of MLP Classifier w/ Autoencoder = 77.79%
Accuracy of MLP Classifier w/ UMAP = 82.97%
Accuracy of MLP Classifier w/ SVD = 89.92%

Overall, we can see that the performance of the model does suffer when utilizing dimensionality reduction. The best results were found when using SVD for dimensionality reduction, which achieved an accuracy of 89.92%, but this still fell short of the original accuracy of 90.46%. The significance of the accuracy loss is dependent on the dimensionality technique utilized, but in the case of SVD, the loss is not significant.

Part 3: Clustering using Both Image and Text

Question 26

Try to construct various text queries regarding types of Pokemon (such as “type: Bug”, “electric type Pokemon”, or “Pokemon 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 used
- The title of the 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.

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

Flygon
Type 1: Ground
Type 2: Dragon

Type: type: Grass, Probability: 0.4815
Type: type: Dragon, Probability: 0.2816
Type: type: Fairy, Probability: 0.0555
Type: type: Psychic, Probability: 0.0513
Type: type: Bug, Probability: 0.0305

Corviknight
Type 1: Flying
Type 2: Steel

Type: type: Dark, Probability: 0.2250
Type: type: Psychic, Probability: 0.2184
Type: type: Ghost, Probability: 0.1528
Type: type: Poison, Probability: 0.1252
Type: type: Grass, Probability: 0.0510

Oranguru
Type 1: Normal
Type 2: Psychic

Type: type: Ghost, Probability: 0.3831
Type: type: Psychic, Probability: 0.1612
Type: type: Rock, Probability: 0.0885
Type: type: Grass, Probability: 0.0863
Type: type: Poison, Probability: 0.0744

Cyndaquil
Type 1: Fire
Type 2:

Type: type: Grass, Probability: 0.3208
Type: type: Fire, Probability: 0.2393
Type: type: Psychic, Probability: 0.0830
Type: type: Dragon, Probability: 0.0798
Type: type: Ghost, Probability: 0.0773

Riolu
Type 1: Fighting
Type 2:

Type: type: Electric, Probability: 0.1361
Type: type: Psychic, Probability: 0.1182
Type: type: Ice, Probability: 0.0955
Type: type: Ghost, Probability: 0.0924
Type: type: Dark, Probability: 0.0765

Lurantis
Type 1: Grass
Type 2:

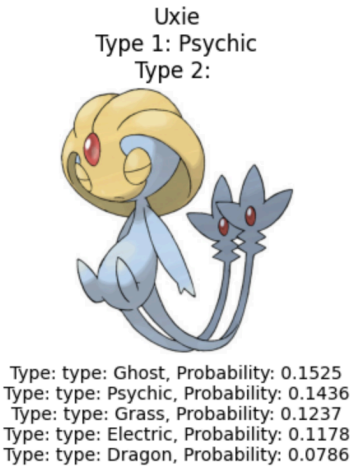
Type: type: Fairy, Probability: 0.2182
Type: type: Poison, Probability: 0.1748
Type: type: Psychic, Probability: 0.1664
Type: type: Dragon, Probability: 0.1071
Type: type: Grass, Probability: 0.0899

Woobat
Type 1: Psychic
Type 2: Flying

Type: type: Ghost, Probability: 0.2702
Type: type: Grass, Probability: 0.0988
Type: type: Psychic, Probability: 0.0923
Type: type: Dark, Probability: 0.0890
Type: type: Dragon, Probability: 0.0853

Bounsweet
Type 1: Grass
Type 2:

Type: type: Grass, Probability: 0.6671
Type: type: Psychic, Probability: 0.1034
Type: type: Poison, Probability: 0.0792
Type: type: Bug, Probability: 0.0495
Type: type: Fairy, Probability: 0.0219



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.