Project 2: Data Representations and Clustering

Tania Rajabally
UID: 806153219

PART 1 - CLUSTERING ON TEXT DATA

QUESTION 1:

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

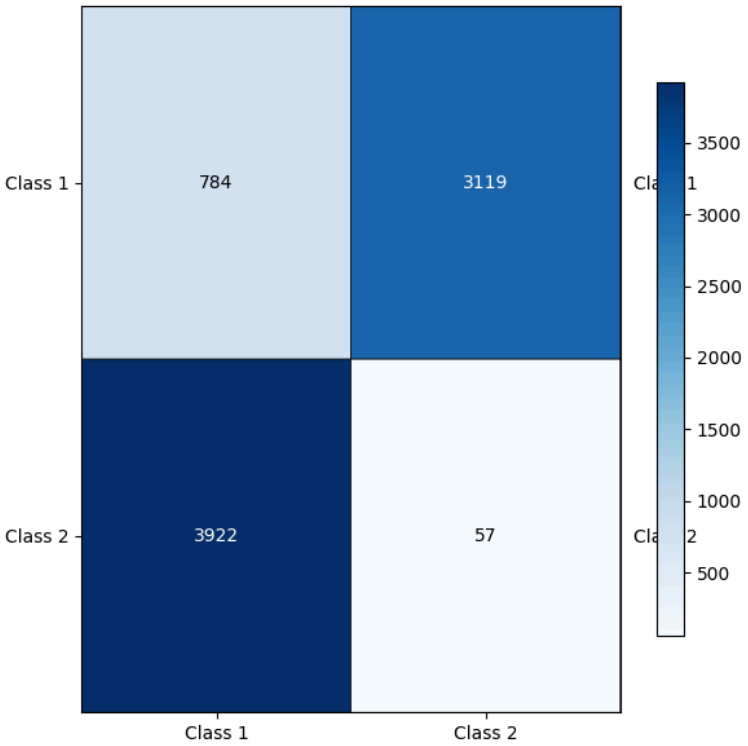
The dimensions of the TF-IDF matrix is (7882, 23522)

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 table is:

[[784 3119]
 [3922 57]]



The contingency table of the clustering result is as found above along with the plot to visualize the matrix. Generally the contingency matrix is square-shaped because the number of clusters is equivalent to the number of labels we have divided the categories into. But, there is a possibility that there is a category in the data which does not have any mapped label or vice versa. In that case, we may not get a square shaped matrix. This is a rare case but it is possible.

QUESTION 3:

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

The 5 clustering measures are as below:

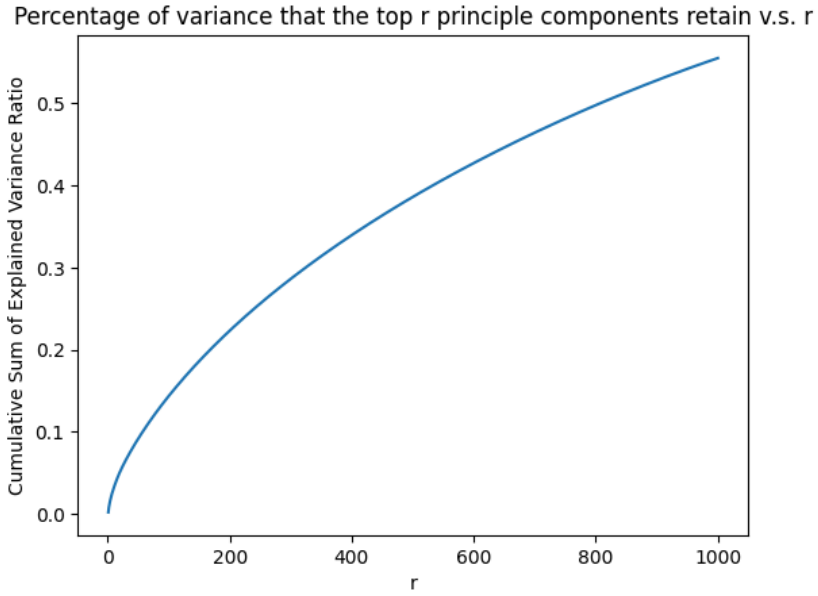
Homogeneity: 0.559685

Completeness: 0.575387

V-measure: 0.567427
Adjusted Rand-Index: 0.618695
Adjusted Mutual Information Score: 0.567387

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.

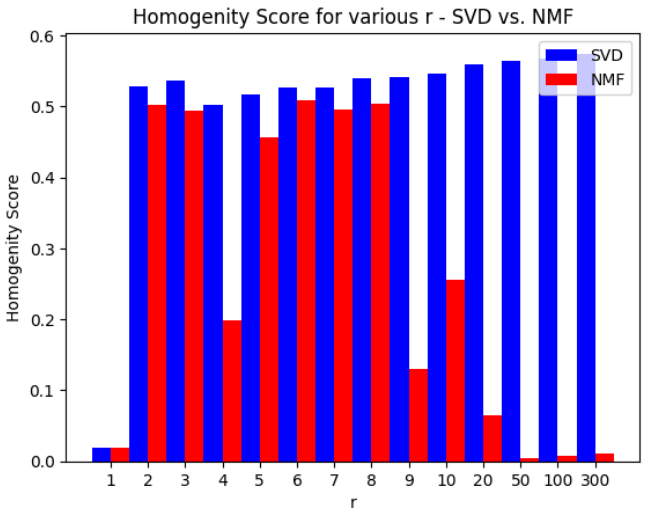


Please refer to the figure above for the plot. We see that as we increase the value of r, the explained variance ratio increases as well.

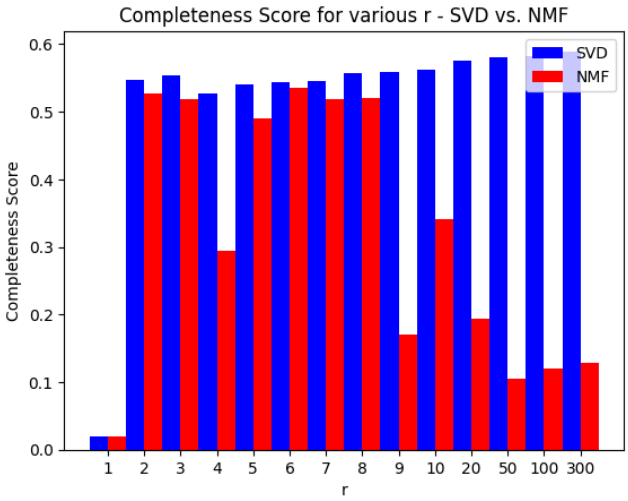
QUESTION 5:

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.

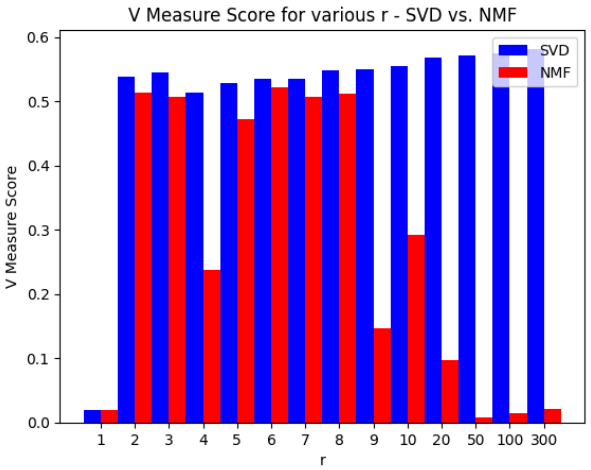
Homogeneity Score for SVD and NMF:



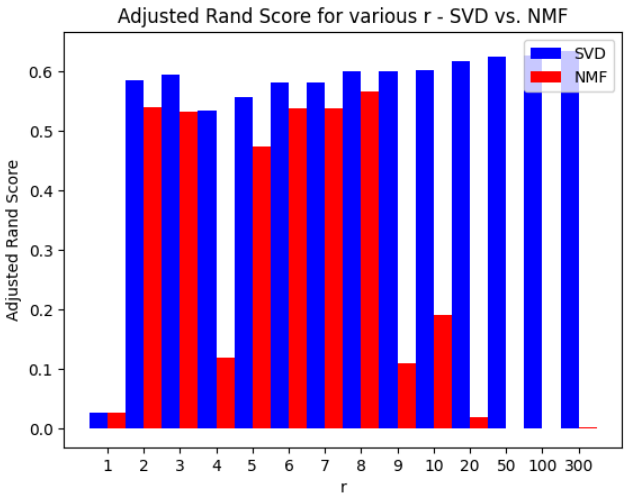
Completeness Score for SVD and NMF:



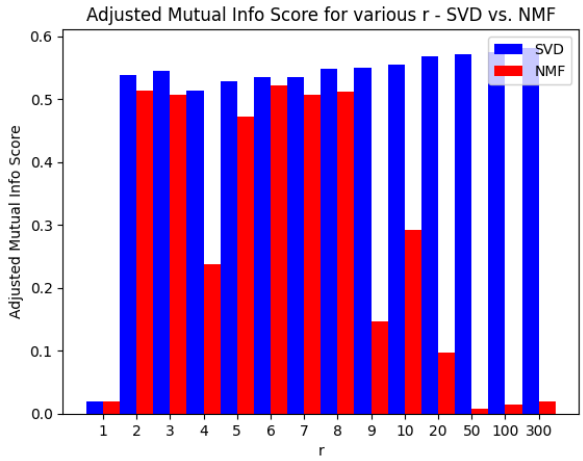
V Measure Score for SVD and NMF:



Adjusted Rand Score for SVD and NMF:



Adjusted Mutual Info Score for SVD and NMF:



From the above graphs, we can see that the good value of r is as below:
SVD = 300
NMF = 6

QUESTION 6:

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

For both SVD and NMF, as we increase the value of r, the variance retained in the reduced dimensions increases. Initially, when we increase the value of r, the clustering performance tends to improve as more information is preserved. However, beyond a certain point, increasing r may lead to overfitting or noise amplification, resulting in poorer clustering performance. This drop is observed because it becomes a high dimensional matrix. At higher values of r, more noise and irrelevant features are captured. The behavior of the k-means depends on the dimensionality of the data and the clustering task can also contribute to the non-monotonic trend. Kmeans using euclidean distance and this measure is not good for high dimensional space. Choosing an appropriate value of r involves balancing the trade-off between retaining sufficient information for clustering while avoiding overfitting or noise amplification

QUESTION 7:

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

Kmeans on Sparse Data:
Homogeneity: 0.559685
Completeness: 0.575387
V-measure: 0.567427
Adjusted Rand-Index: 0.618695
Adjusted Mutual Information Score: 0.567387

Kmeans on SVD with r=300
Homogeneity: 0.5744113629390132
Completeness: 0.5892858802537996
V-measure: 0.5817535576669169
Adjusted Rand-Index: 0.6343636646220453
Adjusted Mutual Information Score: 0.581714777252691

Kmeans on NMF with r=6
Homogeneity: 0.5089511512346101

Completeness: 0.53589420450147
V-measure: 0.5220752924318337
Adjusted Rand-Index: 0.5376953934942157
Adjusted Mutual Information Score: 0.5220304090691056

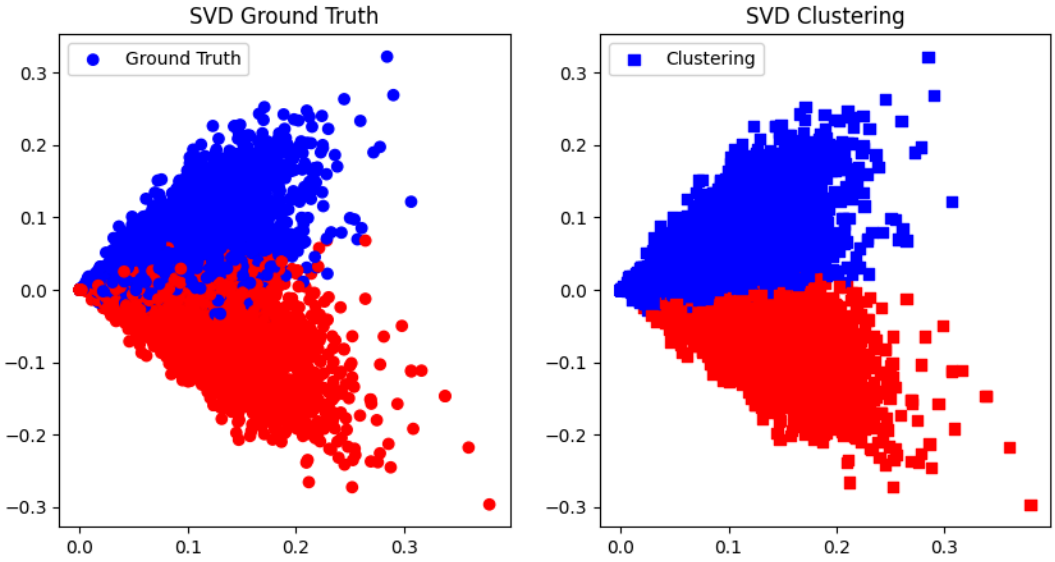
We can see that SVD with r=300 has performed better than NMF with r=6. Also, for sparse data and SVD with r=300, the measure scores are pretty similar with the SVD values just slightly higher

QUESTION 8:

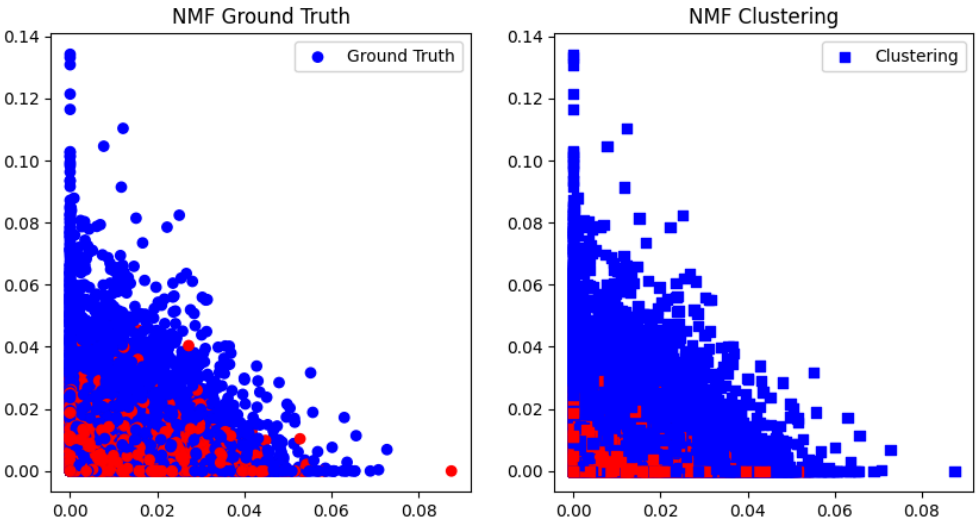
Visualize the clustering results for:

- **SVD with your optimal choice of r for K-Means clustering;**
- **NMF with your choice of r for K-Means clustering.**

SVD with r=300:



NMF with r=6:



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?

We can see that Kmeans assumes that the clusters are isotropic in nature. We notice that NMF shows uneven cluster sizes. Even SVD shows uneven cluster sizes but it is better than NMF. Both show unequal variances for the clusters. For SVD, both the clusters are equally distributed. For NMF, one cluster is very compact and the other one is spread out. For both, there is an overlap of clusters which means that there is a smaller euclidean distance between the centroids of the clusters which means low performance and hence this is not very ideal.

<u>QUESTION 10:</u>

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).

SVD:

Best value of r for SVD: 100 , with the average value of 5 metrics: 0.3128069316335742

The settings we chose for SVD is r=100 as it has the highest average value for all 5 metrics.

Below are the five clustering metrics:

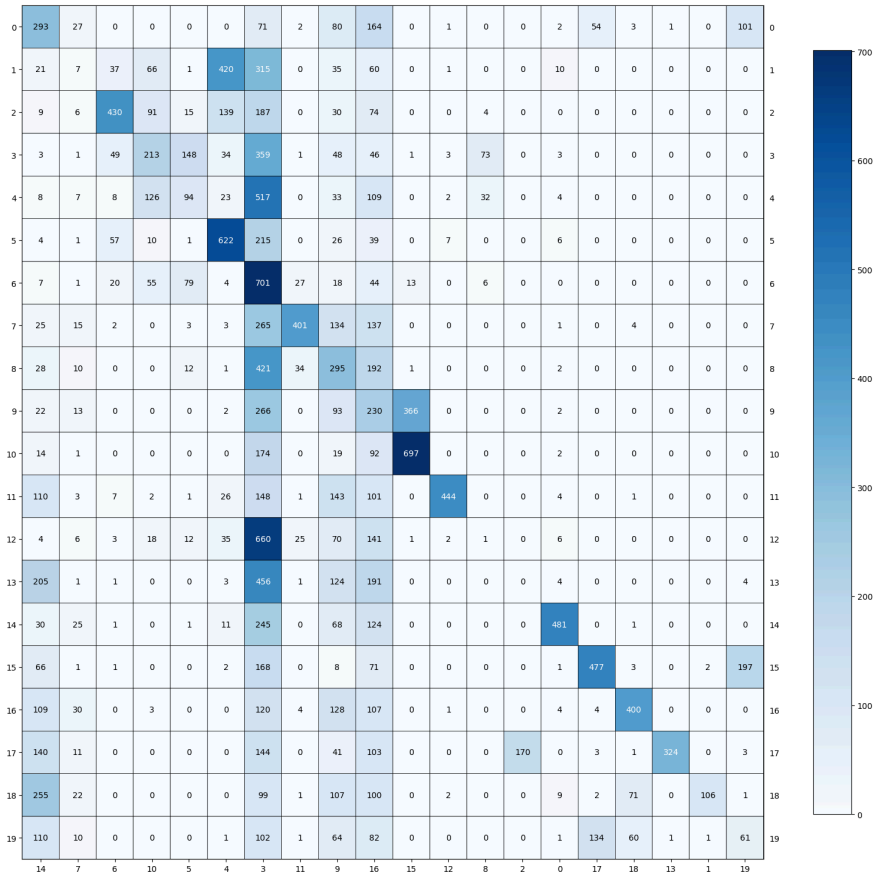
Homogeneity score for all categories using SVD is: 0.3307545340133053

Completeness score for all categories using SVD is: 0.39998464796440997

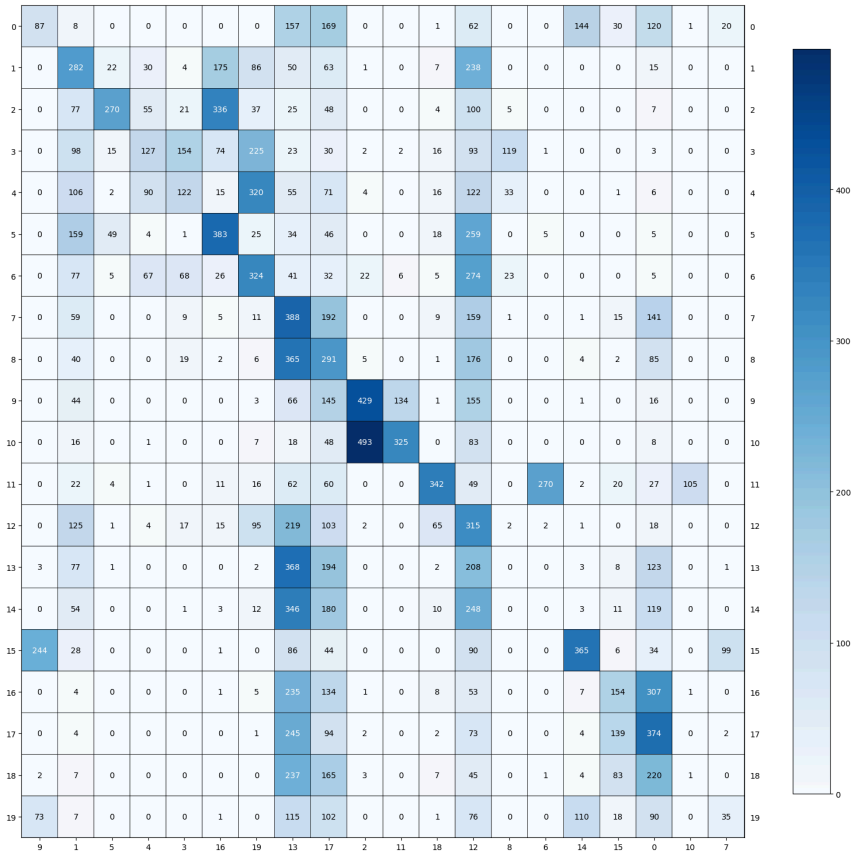
V Measure score for all categories using SVD is: 0.3620901659929847

Adjusted Rand score for all categories using SVD is: 0.1113858195844356

Adjusted Mutual Info score for all categories using SVD is: 0.35981949061273566

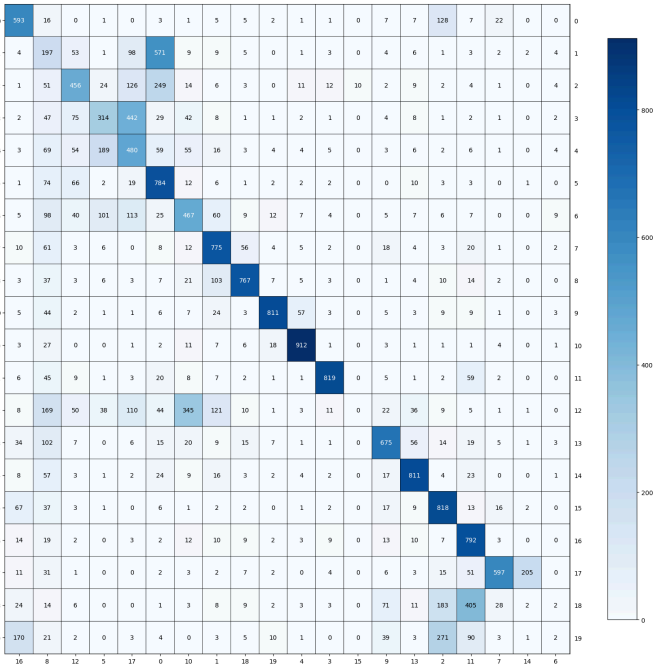


NMF:
Best value of r for NMF: 9 , with the average value of 5 metrics: 0.2692050158980476
The settings we chose for NMF is r=9 as it has the highest average value for all 5 metrics.
Below are the five clustering metrics:
Homogeneity score for all categories using NMF is: 0.29248781037820104
Completeness score for all categories using NMF is: 0.3331444910454053
V Measure score for all categories using NMF is: 0.3114951146342911
Adjusted Rand score for all categories using NMF is: 0.0997830764438181
Adjusted Mutual Info score for all categories using NMF is: 0.30911458698852257



<u>QUESTION 11:</u>
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 this spec.
Report the permuted contingency matrix and the five clustering evaluation metrics for the different combinations (6 combinations).

5 components and cosine metric:
Homogeneity score for all categories using 5 components and cosine metrics is: 0.5531525085510313
Completeness score for all categories using 5 components and cosine metrics is: 0.5829782139137853
V Measure score for all categories using 5 components and cosine metrics is: 0.5676738690023347
Adjusted Rand score for all categories using 5 components and cosine metrics is: 0.42998284859282154
Adjusted Mutual Info score for all categories using 5 components and cosine metrics is:
0.5662283570153432



5 components and euclidean metric:

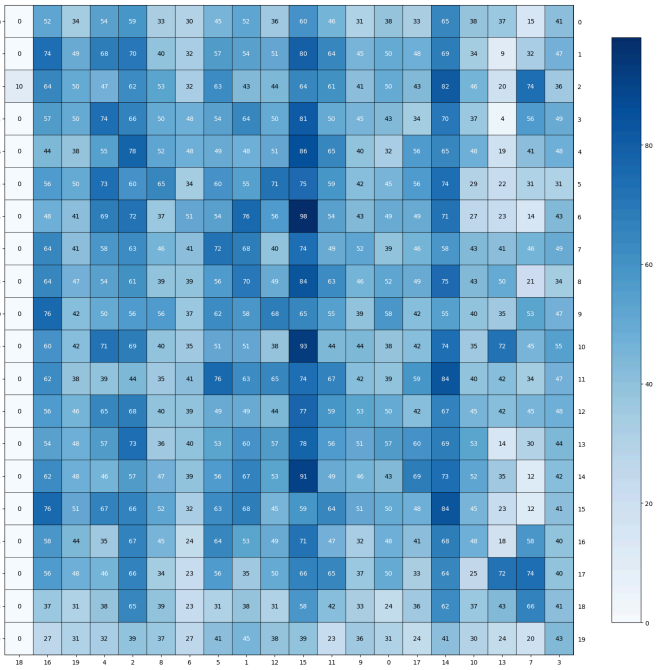
Homogeneity score for all categories using 5 components and euclidean metrics is: 0.007891363023388595

Completeness score for all categories using 5 components and euclidean metrics is: 0.008069457254962472

V Measure score for all categories using 5 components and euclidean metrics is: 0.007979416532494715

Adjusted Rand score for all categories using 5 components and euclidean metrics is: 0.0010915638250219436

Adjusted Mutual Info score for all categories using 5 components and euclidean metrics is: 0.004738753061765688



20 components and cosine metric:

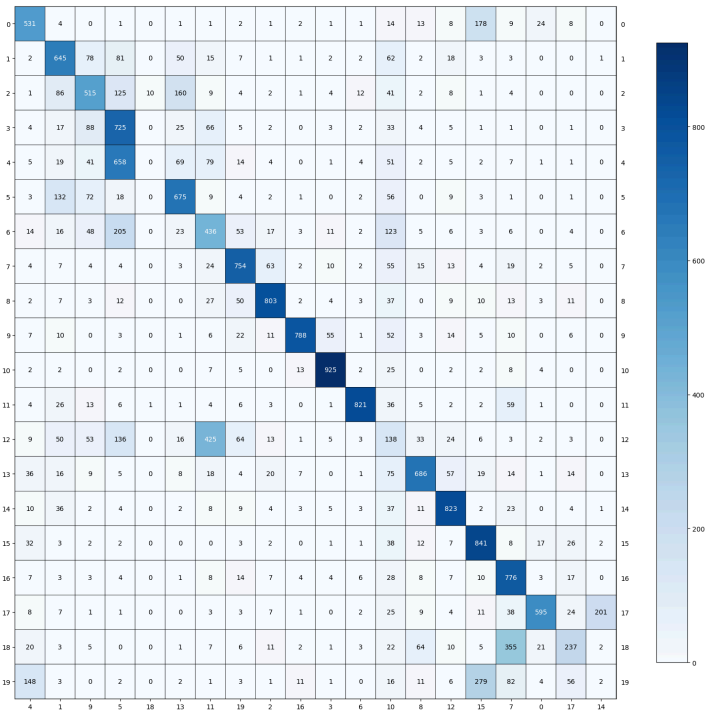
Homogeneity score for all categories using 20 components and cosine metrics is: 0.5749235906774

Completeness score for all categories using 20 components and cosine metrics is: 0.5975831997171487

V Measure score for all categories using 20 components and cosine metrics is: 0.5860344378803356

Adjusted Rand score for all categories using 20 components and cosine metrics is: 0.46004028211510756

Adjusted Mutual Info score for all categories using 20 components and cosine metrics is: 0.5846681328926362



20 components and euclidean metric:

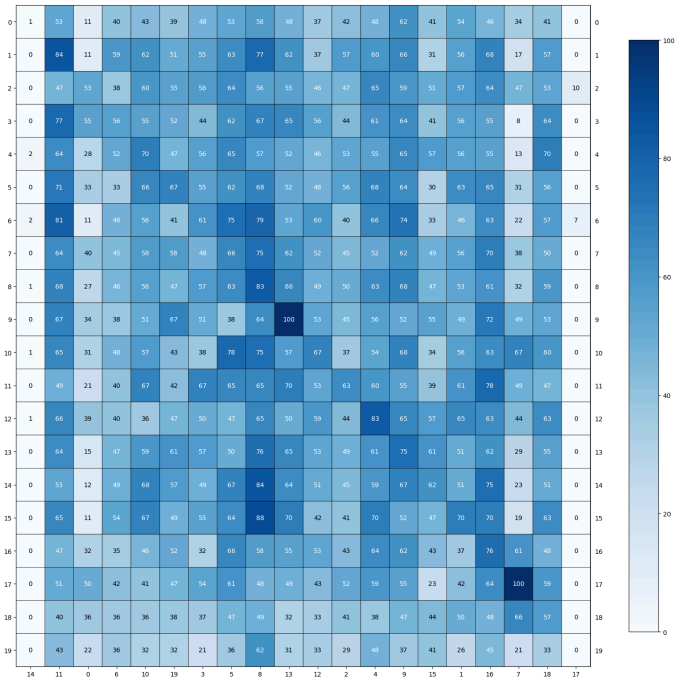
Homogeneity score for all categories using 20 components and euclidean metrics is: 0.007837497040642464

Completeness score for all categories using 20 components and euclidean metrics is: 0.00813689297772979

V Measure score for all categories using 20 components and euclidean metrics is: 0.007984389333130849

Adjusted Rand score for all categories using 20 components and euclidean metrics is: 0.001057509315495261

Adjusted Mutual Info score for all categories using 20 components and euclidean metrics is: 0.004707589930186415



200 components and cosine metric:

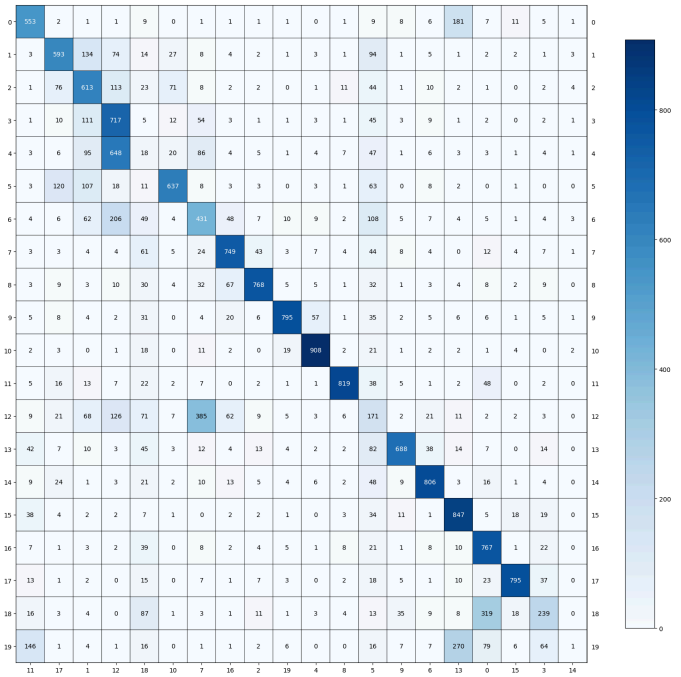
Homogeneity score for all categories using 200 components and cosine metrics is: 0.5834080957166907

Completeness score for all categories using 200 components and cosine metrics is: 0.6009379272213545

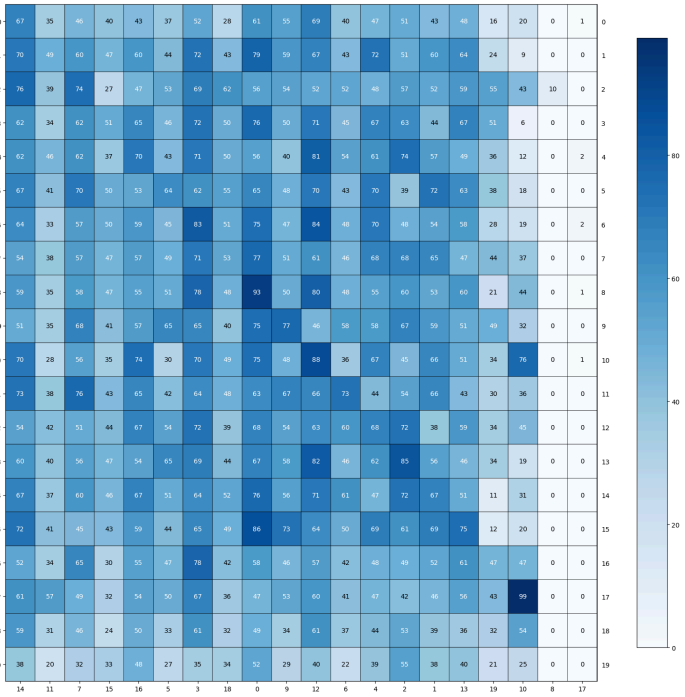
V Measure score for all categories using 200 components and cosine metrics is: 0.5920432795382227

Adjusted Rand score for all categories using 200 components and cosine metrics is: 0.47377099376076715

Adjusted Mutual Info score for all categories using 200 components and cosine metrics is: 0.5906969055771208



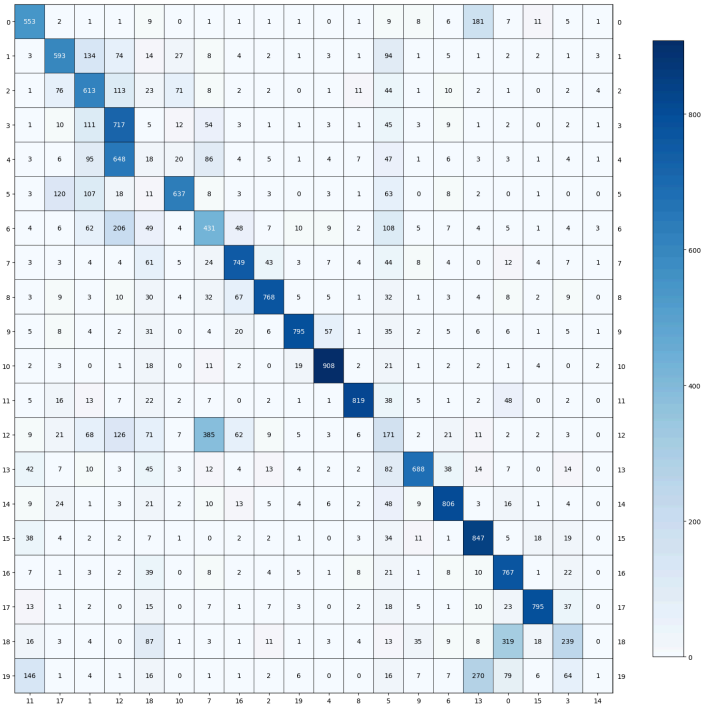
200 components and euclidean metric:
Homogeneity score for all categories using 200 components and euclidean metrics is:
0.007532964477482667
Completeness score for all categories using 200 components and euclidean metrics is:
0.007836776690624627
V Measure score for all categories using 200 components and euclidean metrics is:
0.007681867870480117
Adjusted Rand score for all categories using 200 components and euclidean metrics is:
0.001022975641448769
Adjusted Mutual Info score for all categories using 200 components and euclidean metrics is:
0.00442812370839427



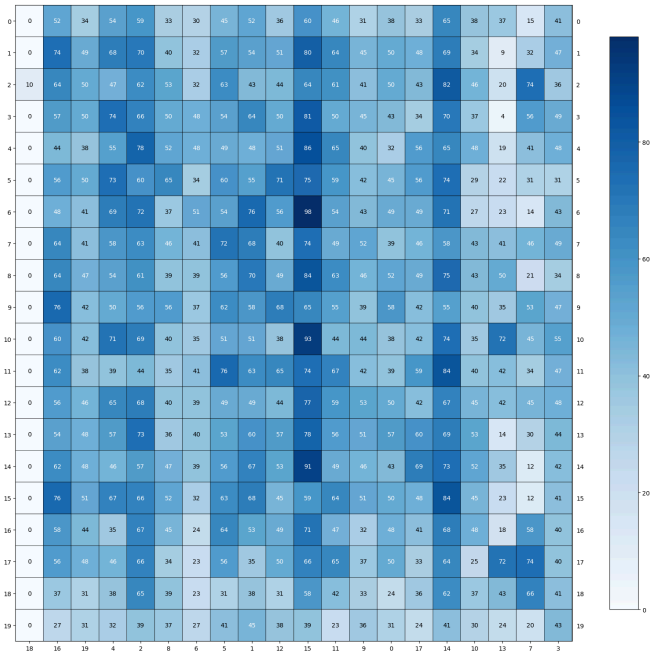
Best value of r for Euclidean UMAP (according to avg. metric): 5 , avg. value of 5 metrics:
0.005954110739526682
Best value of r for Cosine UMAP (according to avg. metric): 200 , avg. value of 5 metrics:
0.5681714403628312

<u>QUESTION 12:</u>
Analyze the contingency matrices. Which setting works best and why? What about for each metric choice?
We can see that for Euclidean UMAP, the contingency matrix is non diagonal and spread out. Whereas for cosine UMAP, some of the ground truth clusters are being mapped to a single cluster and we have a diagonal. In general we can see that the cosine with different components is performing way better than euclidean as a metric. Cosine has a diagonal which means most of the clusters are mapped properly whereas this is not the case for euclidean.
We see that for cosine the best values are found for r=200.
Homogeneity score for all categories using 200 components and cosine metrics is: 0.5834080957166907
Completeness score for all categories using 200 components and cosine metrics is: 0.6009379272213545

V Measure score for all categories using 200 components and cosine metrics is: 0.5920432795382227
Adjusted Rand score for all categories using 200 components and cosine metrics is: 0.47377099376076715
Adjusted Mutual Info score for all categories using 200 components and cosine metrics is: 0.5906969055771208



We can see that for euclidean the best value is found for r=5.
Homogeneity score for all categories using 5 components and euclidean metrics is: 0.007891363023388595
Completeness score for all categories using 5 components and euclidean metrics is: 0.008069457254962472
V Measure score for all categories using 5 components and euclidean metrics is: 0.007979416532494715
Adjusted Rand score for all categories using 5 components and euclidean metrics is: 0.0010915638250219436
Adjusted Mutual Info score for all categories using 5 components and euclidean metrics is: 0.004738753061765688



Comparing both euclidean and cosine, we see that the best metrics are found for r=200 using cosine with an average of the 5 metrics as 0.5681714403628312.

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.

The best results for each of the representation learning techniques are as below:

Sparse TF-IDF:
Homogeneity: 0.327942
Completeness: 0.375275
V-measure: 0.350016
Adjusted Rand-Index: 0.116144
Adjusted Mutual Information Score: 0.347767

SVD: best value of r = 100
Homogeneity score for all categories using SVD is: 0.3307545340133053
Completeness score for all categories using SVD is: 0.39998464796440997
V Measure score for all categories using SVD is: 0.3620901659929847
Adjusted Rand score for all categories using SVD is: 0.1113858195844356
Adjusted Mutual Info score for all categories using SVD is: 0.35981949061273566

NMF: best value of r = 9
Homogeneity score for all categories using NMF is: 0.29248781037820104
Completeness score for all categories using NMF is: 0.3331444910454053

V Measure score for all categories using NMF is: 0.3114951146342911
Adjusted Rand score for all categories using NMF is: 0.0997830764438181
Adjusted Mutual Info score for all categories using NMF is: 0.30911458698852257

UMAP: best value of r = 200 with cosine
Homogeneity score for all categories using 200 components and cosine metrics is: 0.5834080957166907
Completeness score for all categories using 200 components and cosine metrics is: 0.6009379272213545
V Measure score for all categories using 200 components and cosine metrics is: 0.5920432795382227
Adjusted Rand score for all categories using 200 components and cosine metrics is: 0.47377099376076715
Adjusted Mutual Info score for all categories using 200 components and cosine metrics is: 0.5906969055771208

From the above results, we can see that UMAP with cosine and value of r=200 has the best metrics as compared to the other 3. This may be possible since UMAP preserves both local and global structure of the data. It can hence capture all types of relationships. It is also capable of identifying non linear relationships of the data. UMAP constructs a graphical representation of the data. Whereas SVD and NMF use linear relationships of the data. We also see that UMAP for cosine performs better than UMAP for euclidean. This is because the cosine distance is not affected by the magnitude of the vectors. It associates clusters based on the angle between the sample points. Hence UMAP performs the best from the 4 representation learning techniques.

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 with ward linkage:
Homogeneity score for all categories using agglomerative clustering with ward linkage is: 0.5540433824883455
Completeness score for all categories using agglomerative clustering with ward linkage is: 0.5942823738231402
V Measure score for all categories using agglomerative clustering with ward linkage is: 0.5734578619986366
Adjusted Rand score for all categories using agglomerative clustering with ward linkage is: 0.41694193019004505
Adjusted Mutual Info score for all categories using agglomerative clustering with ward linkage is: 0.5720177280839943

Agglomerative Clustering with single linkage:
Homogeneity score for all categories using agglomerative clustering with single linkage is: 0.018333549944148025
Completeness score for all categories using agglomerative clustering with single linkage is: 0.3830029417337657
V Measure score for all categories using agglomerative clustering with single linkage is: 0.03499210117512487
Adjusted Rand score for all categories using agglomerative clustering with single linkage is: 0.0005218260367308564

Adjusted Mutual Info score for all categories using agglomerative clustering with single linkage is: 0.02968964336597013

We can see that ward linkage performs much better than single linkage. This might be because single linkage uses the nearest neighbor concept whereas ward linkage uses minimum increase in sum of squares. Single linkage is fast but it does not work well when we have outliers. Single linkage forms long chain like structures whereas ward linkage forms spherical blobs. This is why ward linkage is better.

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.

I have varied the value of epsilon and checked which performs the best for each of the minimum cluster sizes mentioned.

Evaluation metrics for min_cluster_size=20 and epsilon=0.5:
adjusted_rand_score: 0.23190685436147102
adjusted_mutual_info_score: 0.487065021093364
homogeneity_score: 0.4121838885615617
completeness_score: 0.6026334336412247
v_measure_score: 0.48953794268365874
Average metric score: 0.44466542806825604

Evaluation metrics for min_cluster_size=100 and epsilon=0.5:
adjusted_rand_score: 0.20526952868326936
adjusted_mutual_info_score: 0.4963771355150677
homogeneity_score: 0.41889434318913665
completeness_score: 0.6124023418409557
v_measure_score: 0.4974938453243857
Average metric score: 0.446087438910563

Evaluation metrics for min_cluster_size=200 and epsilon=0.1:
adjusted_rand_score: 0.21126361563018553
adjusted_mutual_info_score: 0.4907849089323803
homogeneity_score: 0.41343356969541817
completeness_score: 0.6068582226082836
v_measure_score: 0.49181138800590757
Average metric score: 0.442830340974435

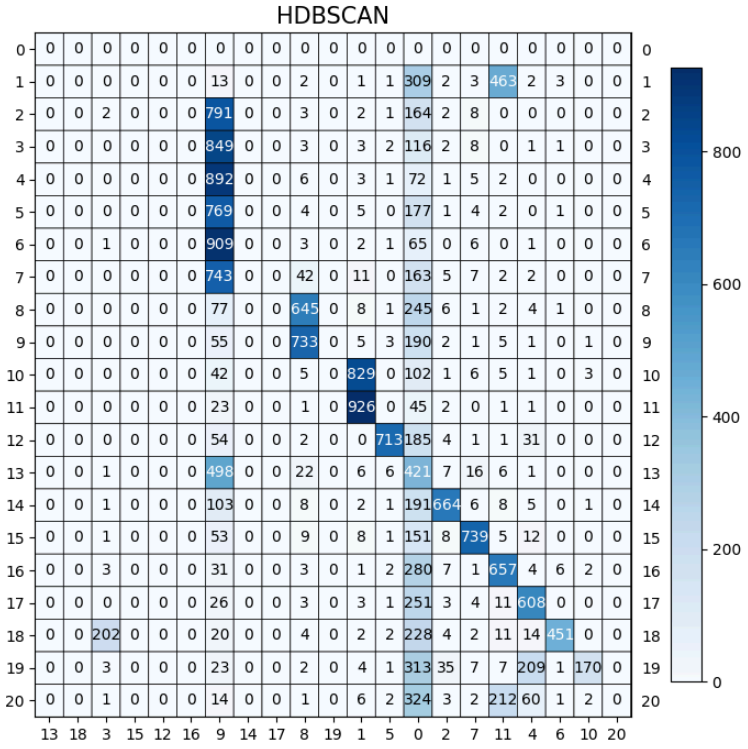
We notice that the best clustering model is for minimum cluster size = 100 and epsilon = 0.5.

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.

We notice that the best clustering model is for minimum cluster size = 100 and epsilon = 0.5. The contingency matrix is as below:



We can see that hdbscan forms 12 clusters and it forms a -1 cluster. We can see that hdbscan forms a lesser number of clusters than the actual categories that we have. This is because it does not take the number of clusters as an input. Instead, it forms clusters on the basis of the density of the feature vectors. -1 signifies all the outliers. The sensitivity to the density of determining the clusters depends on the hyperparameters like epsilon as seen above.

<u>QUESTION 17:</u>

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.

Below are the top 4 models that performed well and the average score of the 5 metrics:

- *UMAP with cosine and n_components=200 and kmeans with 20 clusters -*

Average score of metrics - 0.568015

- *UMAP with cosine and n_components=20 and kmeans with 20 clusters -*

Average score of metrics - 0.561719

- *UMAP with cosine and n_components=5 and kmeans with 20 clusters -*

Average score of metrics - 0.55774

- *UMAP with cosine and n_components=200 and agglomerative clustering with ward linkage with 20 clusters -*

Average score of metrics - 0.546844

QUESTION 18:

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

Various techniques were tried as described below. The results for the same can be found in the notebook.

* Lemmatization was used
* min_df was changed
* the headers and footers from the data were not removed - the headers and footers might contain semantic information.
* For hdbscan, the min_epsilon value and min number of samples can be changed.
* We can also apply Linear Discriminant Analysis for dimensionality reduction.
* We can also try local linear embedding.

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?

The VGG network is trained on a dataset with different classes as targets. Despite this, the features learned by the VGG Network are very abstract and hierarchical representations of the visual information. Lower layers of the network capture low level features such as edges and textures. The higher layers learn more complex and abstract features which are required for classification tasks. These features that are learned are generalizable and transferable over different datasets. This is the reason why the features derived from such a network have discriminative power for a custom dataset. This is because they encode relevant visual information that can be useful for other tasks. Also, transfer learning techniques can make use of these pre-trained features by fine tuning the network on a custom dataset thereby adapting new characteristics and learning from the custom dataset along with retaining information from the original dataset.

QUESTION 20:

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

The helper code first loads the flower photos dataset. We then transform the dataset by resizing, center cropping and normalization to ensure that the data is homogeneous. Then a batch size of 64 is set. We divide the entire dataset into batches each having a size of 64 samples. Then we call feature extract which loads the vgg16 model first. VGG016 has multiple feature layers which include ReLU, Conv2d and MaxPool2d. The image is converted into a one dimensional vector and the first part of the fully connected layer of VGG16 is extracted. The forward method takes an input and gives the feature vector as the output. The training is done in 58 epochs and we obtain 4096 features per sample.

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 images of the dataset are of varied sizes. Hence, we have resized it to 224x224. We resize the image to a square shape of 224 pixels. We then do center crop which crops the image around the center creating a new image with the same dimensions. Together, resizing and center crop ensure that each input image is resized to

a common size (224x224 pixels) and then cropped around its center to produce a consistent input size for the neural network. The VGG Network extracts 4096 features per image sample.

QUESTION 22:

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

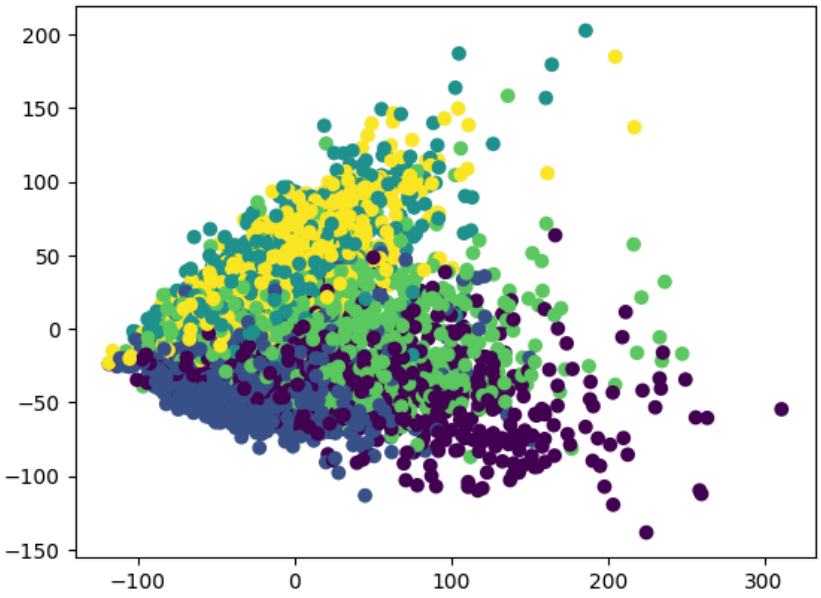
The extracted features from VGG16 are dense, unlike sparse TF-IDF features commonly used in text data. We can find the density by counting the number of non zero entities they have. For TF-IDF, we see that we get a sparse matrix with many non zero elements. But, for VGG16, we notice that all the entities are non zero and hence it is denser as compared to TF-IDF. In summary, while VGG16 extracts dense feature representations for images, TF-IDF generates sparse feature representations for text. The choice between dense and sparse representations depends on the nature of the data and the requirements of the machine learning task.

QUESTION 23:

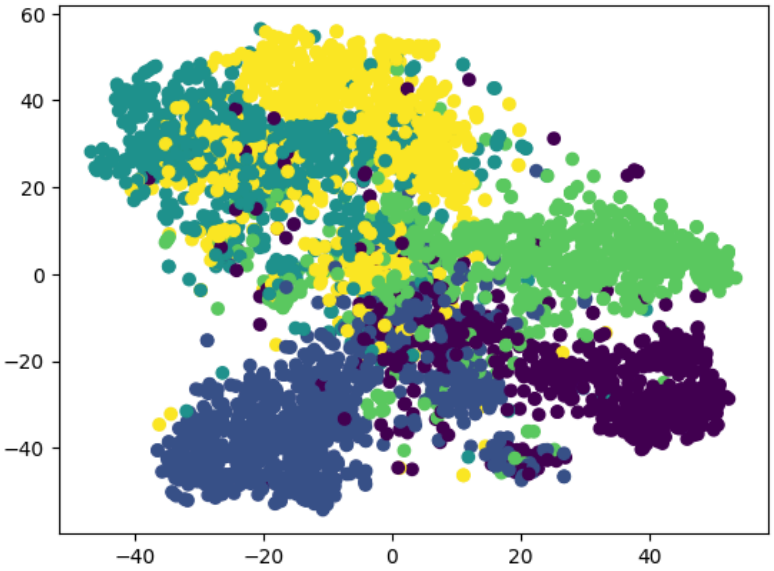
In order to inspect the high-dimensional features, t-SNE is a popular off-the-shelf choice for visualizing Vision features. Map the features you have extracted onto 2 dimensions with t-SNE. Then plot the mapped feature vectors along x and y axes. Color-code the data points with ground-truth labels. Describe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

We can see that t-SNE preserves more information as compared to PCA and hence it is a better choice for visualizing Vision features. t-SNE adopts a probabilistic approach and uses probability to group points such that two close points come from the same probability distribution. Whereas with PCA, it separates points far from each other on the basis of high variance. PCA is a linear dimensionality reduction technique. t-SNE is a non linear dimensionality reduction technique. Therefore t-SNE is generally preferred as it is more robust to outliers and preserves the local structure of the data.

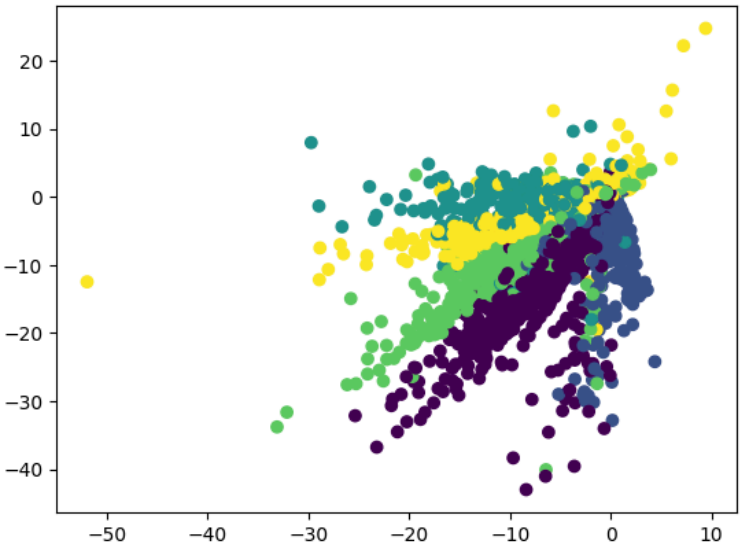
Plot for PCA:



Plot for t-SNE:



Plot for Autoencoder:



<u>QUESTION 24:</u>
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.

Kmeans:
Kmeans Clustering Rand Score: 0.7038367707049935
SVD + Kmeans Clustering Rand Score: 0.7047006252399699
UMAP + Kmeans Clustering Rand Score: 0.8225015094432104
Autoencoder + Kmeans Clustering Rand Score: 0.7322016779512864

Agglomerative Clustering:
Agglomerative Clustering Rand Score: 0.7104411881564593
SVD + Agglomerative Clustering Rand Score: 0.7063276304972139

UMAP + Agglomerative Clustering Rand Score: 0.81895771553846462
Autoencoder + Agglomerative Clustering Rand Score: 0.7224684613630811

HDBSCAN:
For HDBSCAN:
Best value of minimum cluster size is: 5 and minimum number of samples is: 5
Rand Score: 0.4174787953863395
For HDBSCAN with SVD:
Best value of minimum cluster size is: 10 and minimum number of samples is: 5
Rand Score: 0.43632466730980457
For HDBSCAN with UMAP:
Best value of minimum cluster size is: 5 and minimum number of samples is: 10
Rand Score: 0.4572071921534203
For HDBSCAN with Autoencoder:
Best value of minimum cluster size is: 10 and minimum number of samples is: 5
Rand Score: 0.42659947137924864

We can see that Kmeans with UMAP performs the best from all with a rand score of 0.8225015094432104 followed by Agglomerative Clustering with UMAP with a rand score of 0.81895771553846462.

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.

MLP Classifier on the original VGG features are as below:

- Test Accuracy without any dimensionality reduction - 91.41689373297002
- Test Accuracy with SVD - 89.23705722070845
- Test Accuracy with UMAP - 83.65122615803816
- Test Accuracy with autoencoder - 84.46866485013625

We can see that MLP without any dimensionality reduction performs the best from all the above combinations. Without dimensionality reduction, the test accuracy is pretty close to that with SVD. For the remaining models, we can see a drop in test accuracy. For some it is a slight drop whereas for some it is more. The clusters in clustering models are formed on the basis of distance. But in MLP, the model learns the complex features and predicts the classes based on the learning. Therefore MLP is better than clustering.

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 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.

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 are observed due to the model's understanding of textual descriptions and the complexity of the dataset.

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;

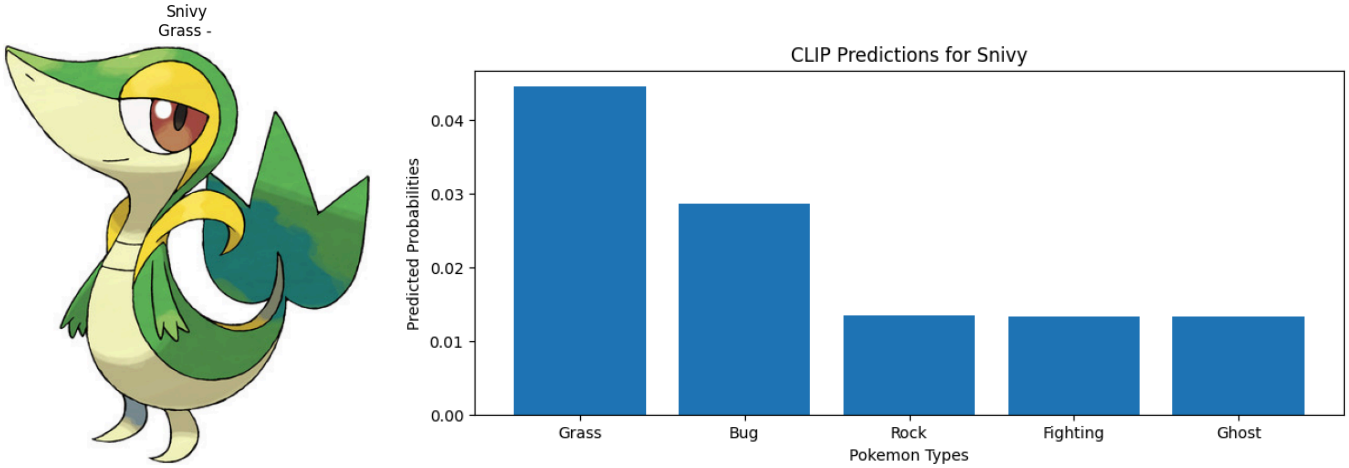
• the five most relevant types predicted by CLIP and their predicted probabilities.

Using only Type 1 as a parameter:

----------Pokemon No 395 ----------

Top five types are: ['Grass', 'Bug', 'Rock', 'Fighting', 'Ghost']

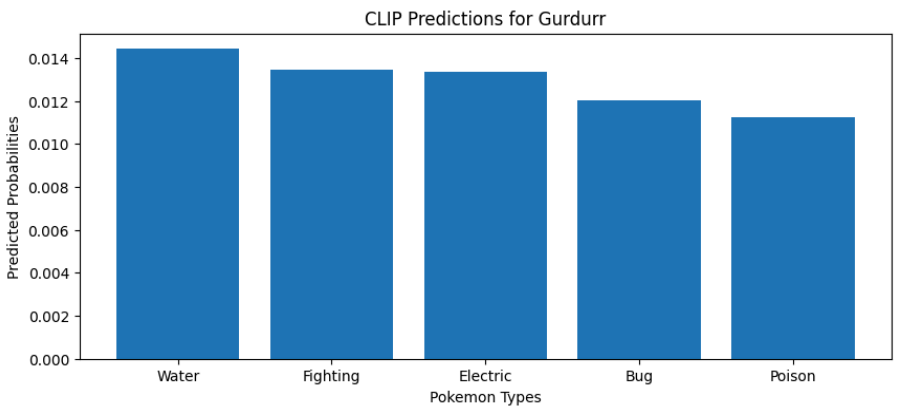
Top five probabilities are: [0.044443645, 0.028661663, 0.01337572, 0.013232502, 0.013226294]



----------Pokemon No 431 ----------

Top five types are: ['Water', 'Fighting', 'Electric', 'Bug', 'Poison']

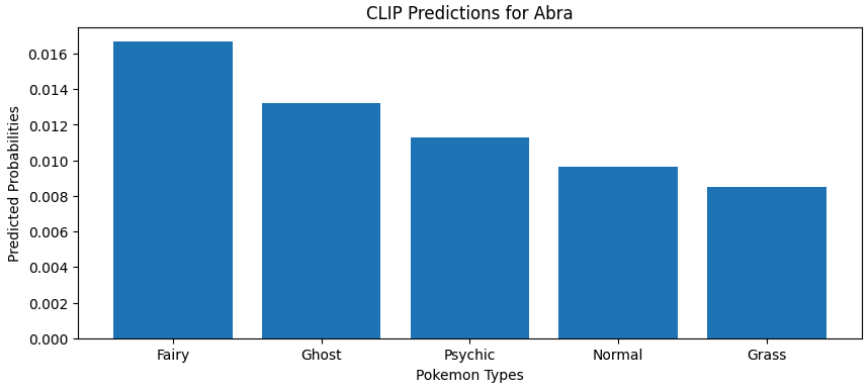
Top five probabilities are: [0.01442239, 0.013468966, 0.013355791, 0.012021065, 0.011262848]



----------Pokemon No 42 ----------

Top five types are: ['Fairy', 'Ghost', 'Psychic', 'Normal', 'Grass']

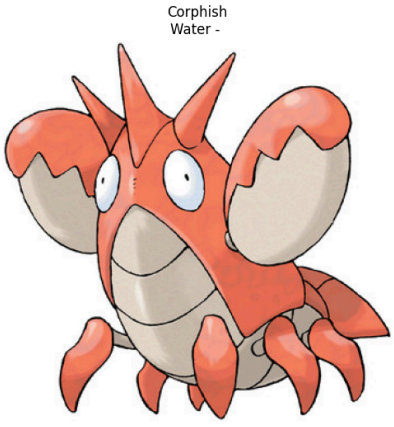
Top five probabilities are: [0.016664894, 0.013219326, 0.011272634, 0.009670248, 0.0085243]

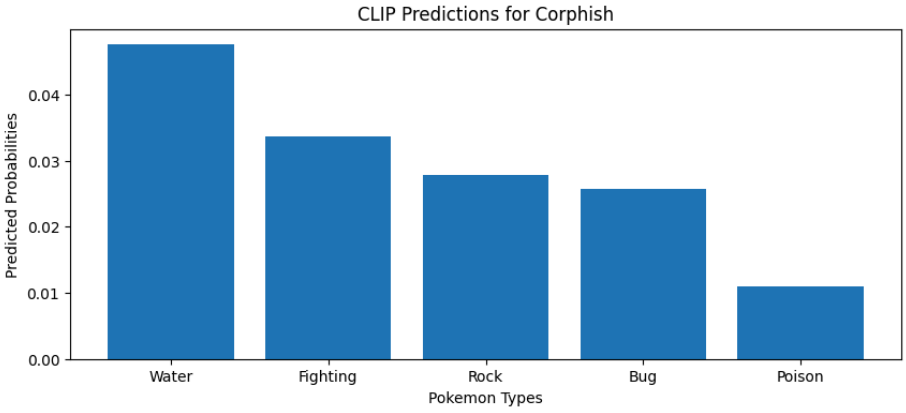


----------Pokemon No 266 ----------

Top five types are: ['Water', 'Fighting', 'Rock', 'Bug', 'Poison']

Top five probabilities are: [0.047561105, 0.033682674, 0.027768934, 0.025689662, 0.0109966565]

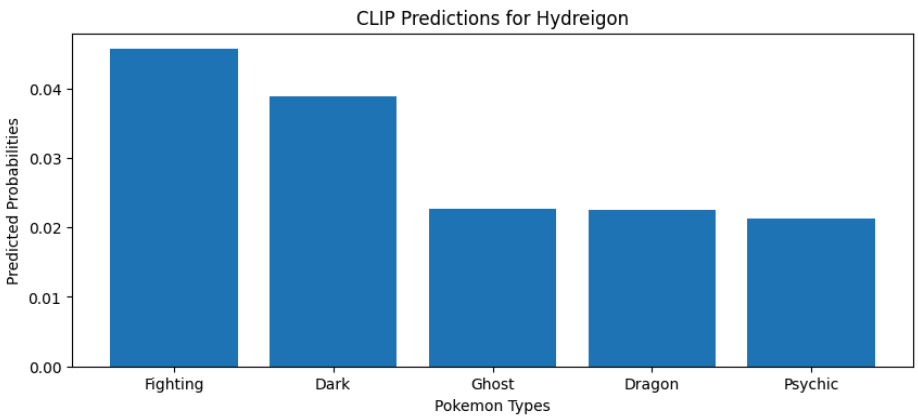




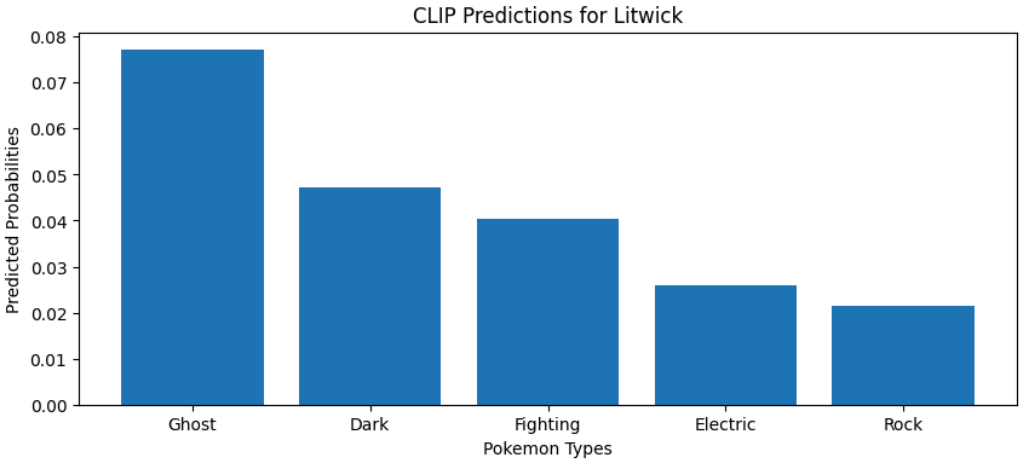
----------Pokemon No 524 ----------

Top five types are: ['Fighting', 'Dark', 'Ghost', 'Dragon', 'Psychic']

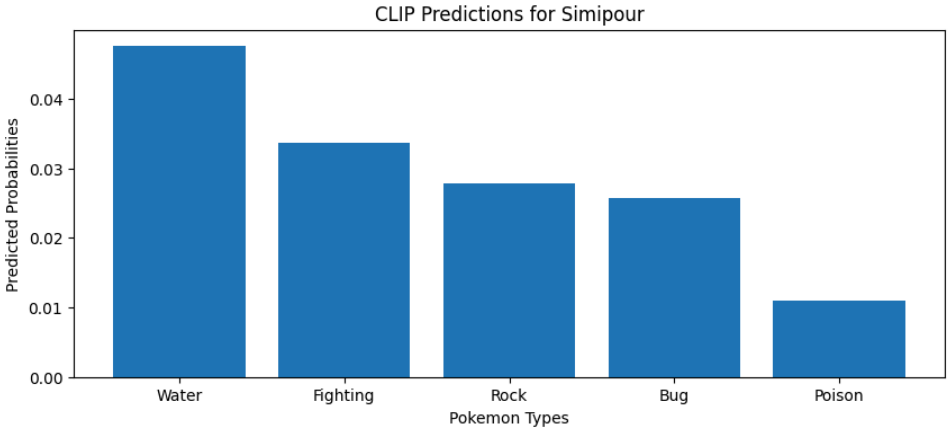
Top five probabilities are: [0.045628317, 0.038835052, 0.022598857, 0.02253438, 0.021266706]



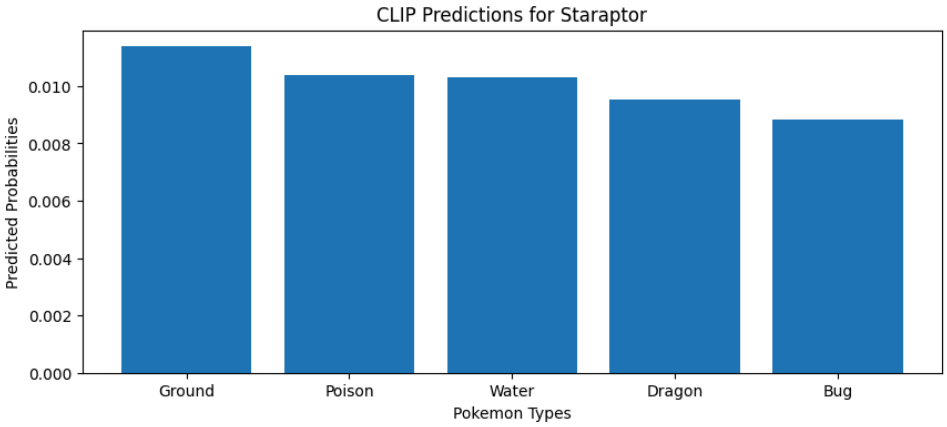
----------Pokemon No 498 ----------
Top five types are: ['Ghost', 'Dark', 'Fighting', 'Electric', 'Rock']
Top five probabilities are: [0.07706201, 0.047104817, 0.040326715, 0.026028544, 0.021373445]



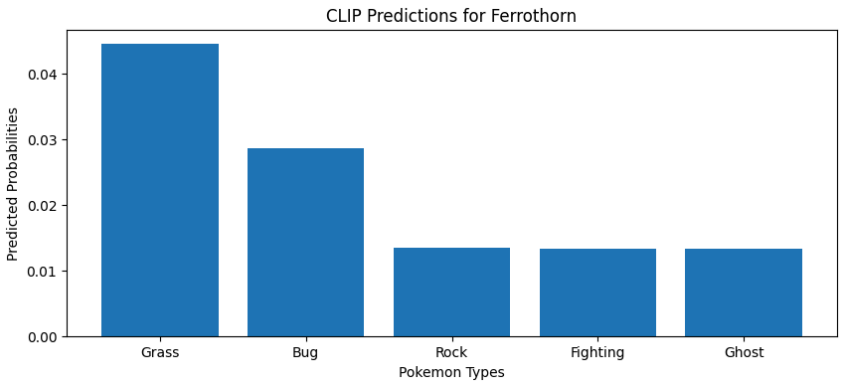
----------Pokemon No 415 ----------
Top five types are: ['Water', 'Fighting', 'Rock', 'Bug', 'Poison']
Top five probabilities are: [0.047561105, 0.033682674, 0.027768934, 0.025689662, 0.0109966565]



----------Pokemon No 311 ----------
Top five types are: ['Ground', 'Poison', 'Water', 'Dragon', 'Bug']
Top five probabilities are: [0.011378028, 0.010374822, 0.010310394, 0.009543379, 0.008818644]

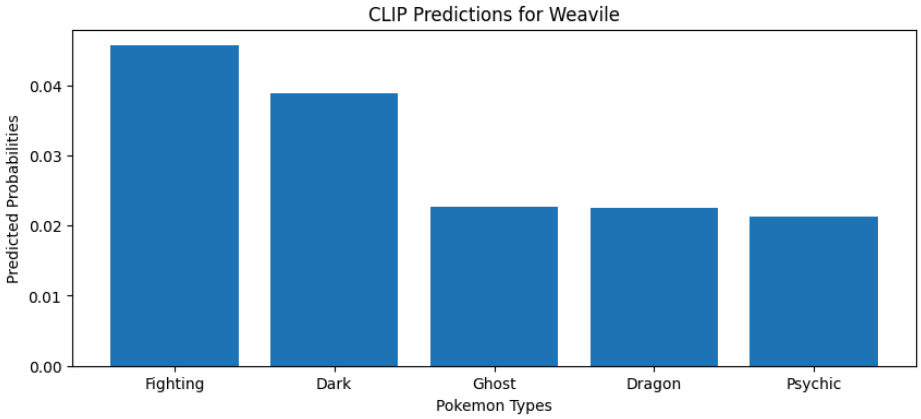


----------Pokemon No 489 ----------
Top five types are: ['Grass', 'Bug', 'Rock', 'Fighting', 'Ghost']
Top five probabilities are: [0.044443645, 0.028661663, 0.01337572, 0.013232502, 0.013226294]

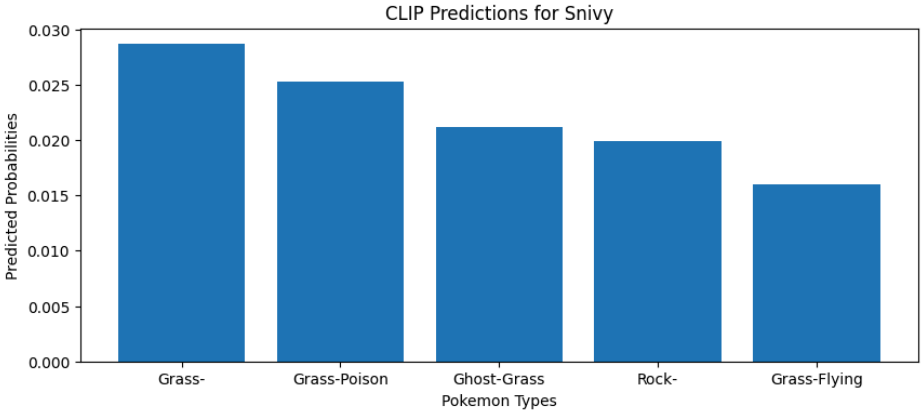


----------Pokemon No 367 ----------

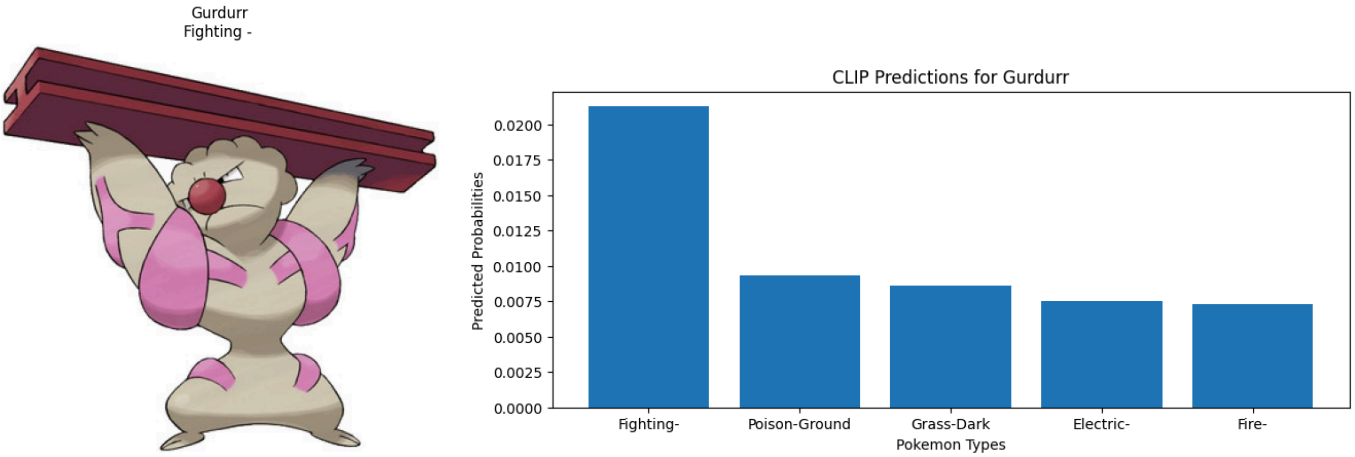
Top five types are: ['Fighting', 'Dark', 'Ghost', 'Dragon', 'Psychic']
Top five probabilities are: [0.045628317, 0.038835052, 0.022598857, 0.02253438, 0.021266706]



Using Type 1 and Type 2 as a parameter:
----------Pokemon No 395 ----------
Top five types are: ['Grass-', 'Grass-Poison', 'Ghost-Grass', 'Rock-', 'Grass-Flying']
Top five probabilities are: [0.028647078, 0.025300492, 0.021156617, 0.01988868, 0.01602355]



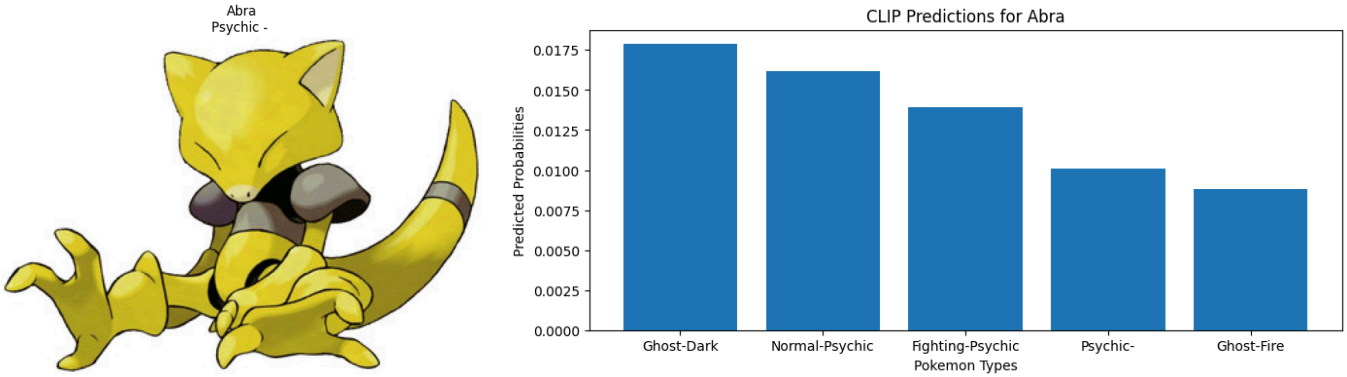
----------Pokemon No 431 ----------
Top five types are: ['Fighting-', 'Poison-Ground', 'Grass-Dark', 'Electric-', 'Fire-']
Top five probabilities are: [0.021253446, 0.009355053, 0.008638209, 0.0075404607, 0.0072647696]



----------Pokemon No 42 ----------

Top five types are: ['Ghost-Dark', 'Normal-Psychic', 'Fighting-Psychic', 'Psychic-', 'Ghost-Fire']

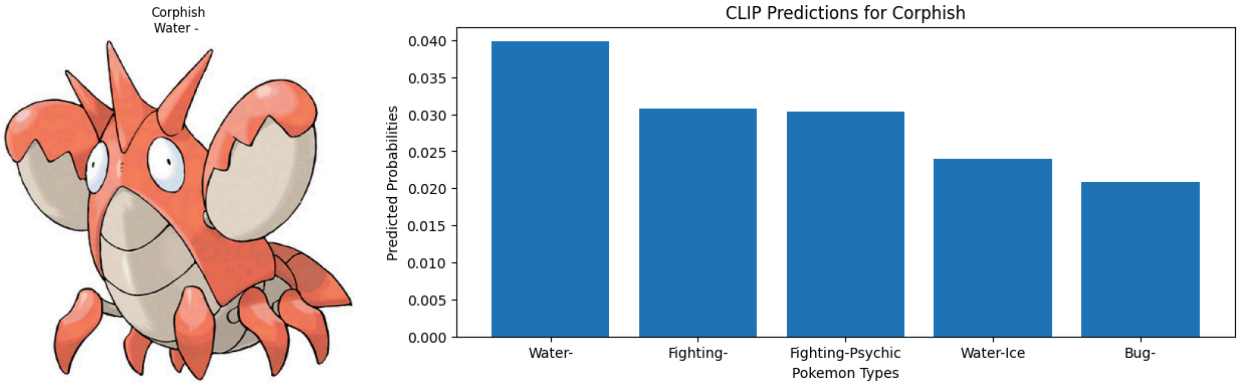
Top five probabilities are: [0.017863266, 0.01619507, 0.013928363, 0.010116057, 0.008807744]



----------Pokemon No 266 ----------

Top five types are: ['Water-', 'Fighting-', 'Fighting-Psychic', 'Water-Ice', 'Bug-']

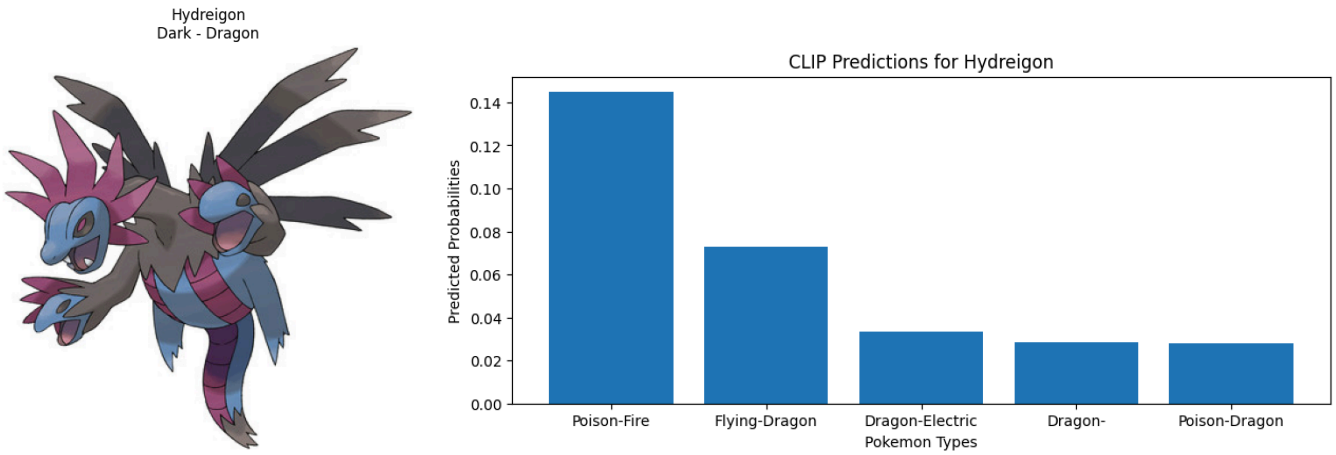
Top five probabilities are: [0.039803855, 0.03081699, 0.030372743, 0.023999106, 0.0208298]



----------Pokemon No 524 ----------

Top five types are: ['Poison-Fire', 'Flying-Dragon', 'Dragon-Electric', 'Dragon-', 'Poison-Dragon']

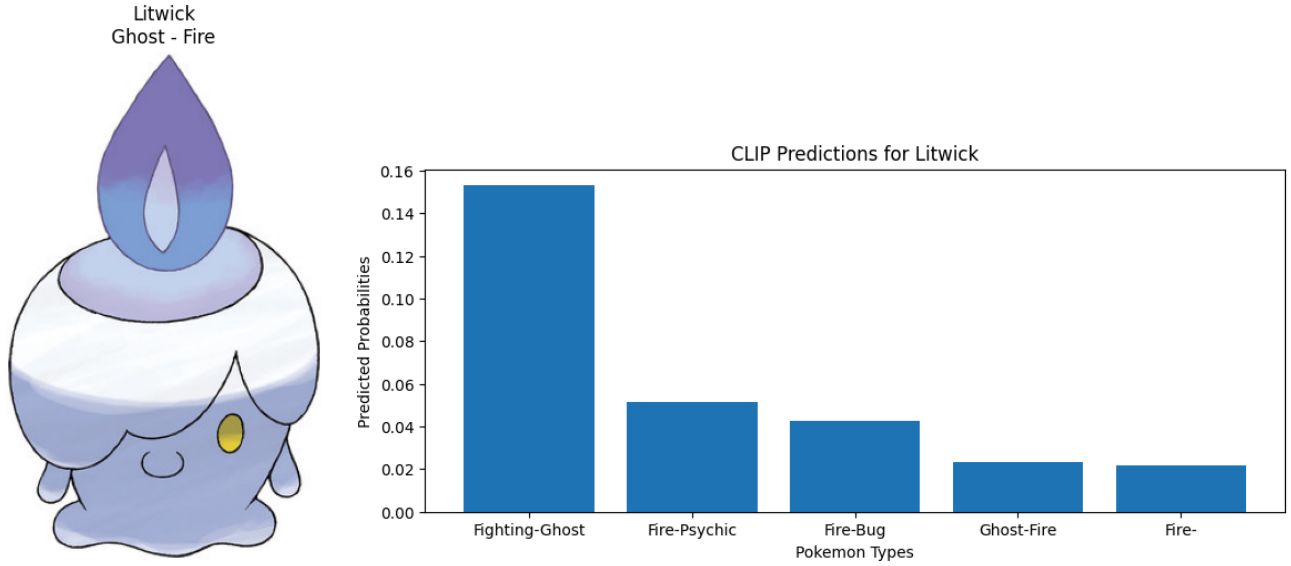
Top five probabilities are: [0.14468619, 0.07263286, 0.03359808, 0.028689658, 0.02820083]



----------Pokemon No 498 ----------

Top five types are: ['Fighting-Ghost', 'Fire-Psychic', 'Fire-Bug', 'Ghost-Fire', 'Fire-']

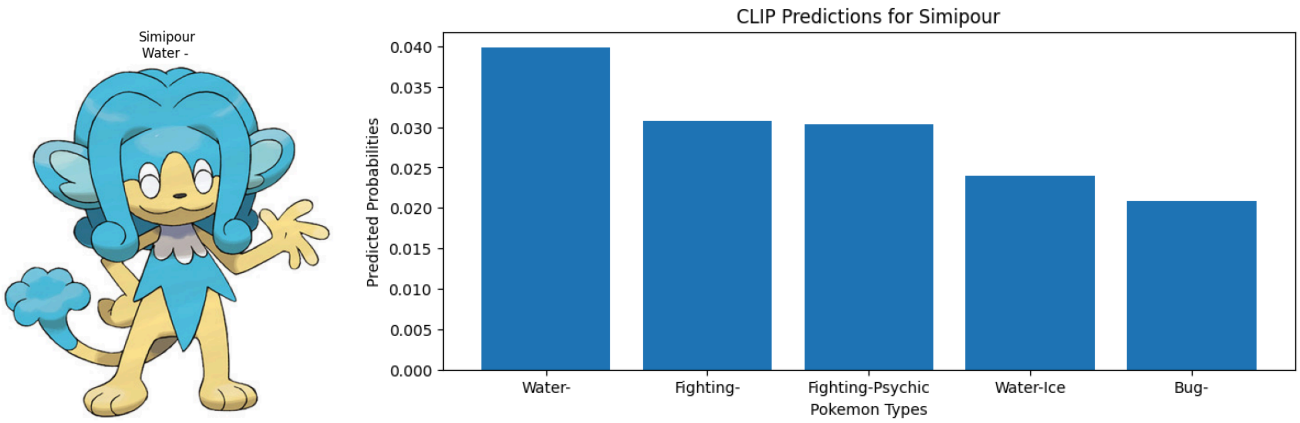
Top five probabilities are: [0.15291385, 0.051707964, 0.042517424, 0.023266807, 0.022008834]



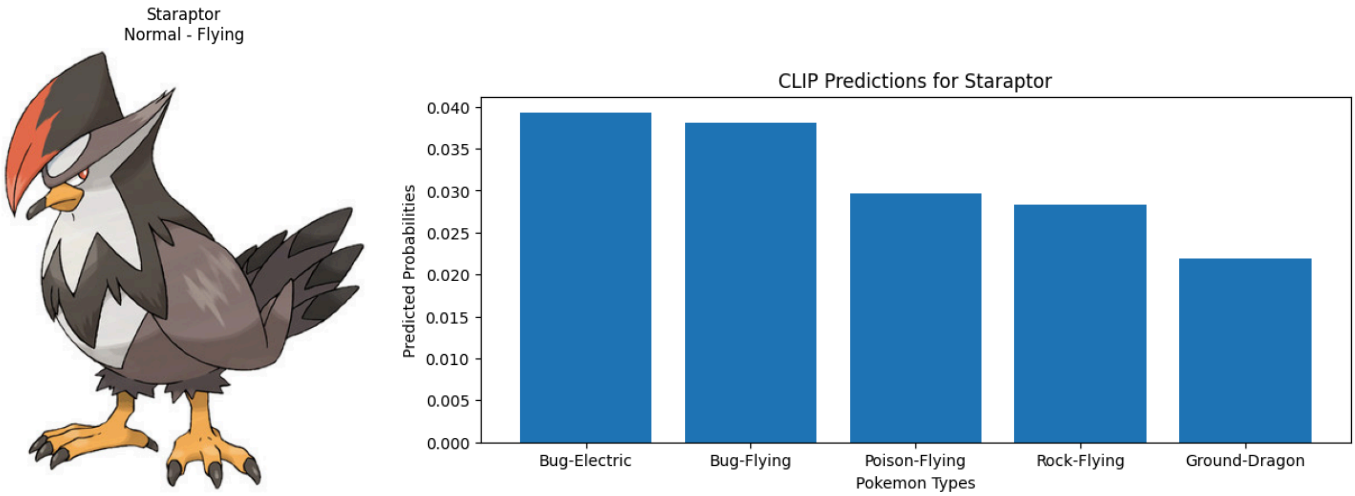
----------Pokemon No 415 ----------

Top five types are: ['Water-', 'Fighting-', 'Fighting-Psychic', 'Water-Ice', 'Bug-']

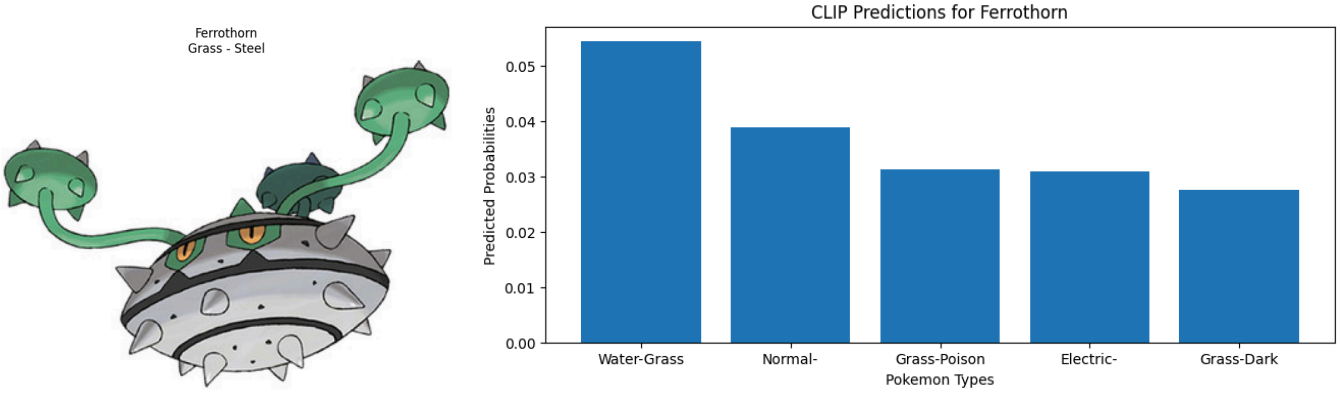
Top five probabilities are: [0.039803855, 0.03081699, 0.030372743, 0.023999106, 0.0208298]



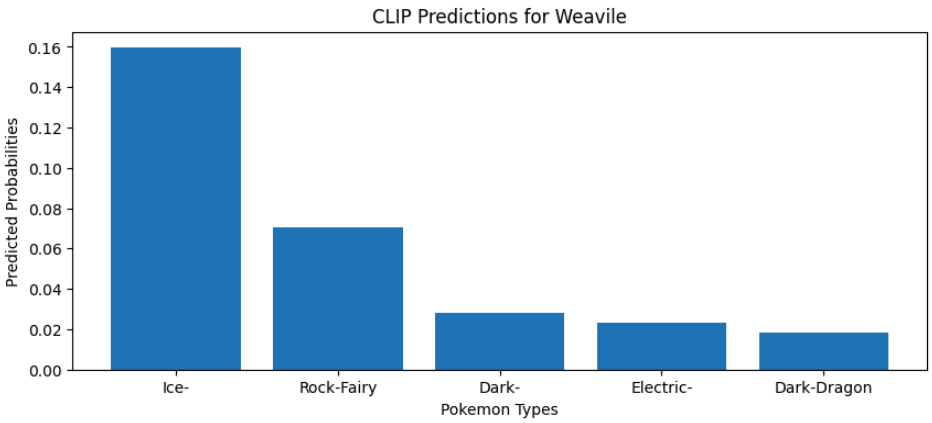
----------Pokemon No 311 ----------
Top five types are: ['Bug-Electric', 'Bug-Flying', 'Poison-Flying', 'Rock-Flying', 'Ground-Dragon']
Top five probabilities are: [0.039233994, 0.038048074, 0.02962701, 0.028336532, 0.021900658]



----------Pokemon No 489 ----------
Top five types are: ['Water-Grass', 'Normal-', 'Grass-Poison', 'Electric-', 'Grass-Dark']
Top five probabilities are: [0.05439731, 0.038959887, 0.031267196, 0.030867457, 0.027516888]



----------Pokemon No 367 ----------
Top five types are: ['Ice-', 'Rock-Fairy', 'Dark-', 'Electric-', 'Dark-Dragon']
Top five probabilities are: [0.15948467, 0.070260204, 0.028377406, 0.02343031, 0.018104127]



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 neighbor- hood 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.

The graph above shows the plot for the Pokemon types Bug, Fire and Grass. We can see that three clusters are formed. They are not very distinct. The bottom left corner depicts a cluster of Bug Pokemon. The top tight blue cluster represents Grass. The spread out red cluster is Fire. We notice that we do not have well defined clusters because there are certain characteristics that are overlapping and hence we can see the overlapping of two clusters and some point being plotted in between. We can see that fire has the most spread out cluster as compared to the other two pokemon types.