Project 2 Data Representations and Clustering

Maya Deshpande, UID: 106076875, email: mayadeshpande@g.ucla.edu
Yujie Wang, UID: 406077623 email: yujie2000@g.ucla.edu
Haochen Zhang, UID: 706071484, email: haochen235@g.ucla.edu

February 6, 2024

1 Github Repo Address

All code can be found here
https://github.com/HaochenZhang717/project2_ece219.git

2 Part 1 - Clustering on Text Data

2.1 Question 1

The shape of the TF-IDF matrix is: (7882, 23522).

2.2 Question 2

Contingency Matrix:
[[799 3104]
[3923 56]]

2.3 Question 3

measure	*score*
Homogeneity Score	0.5555385913621526
Completeness Score	0.5718079023699979
V-measure	0.5635558515124099
Adjusted Rand Index	0.61311871193327
Adjusted Mutual Information Score	0.5635153178210184

Table 1: Five clustering metrics of K-Means.

2.4 Question 4

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

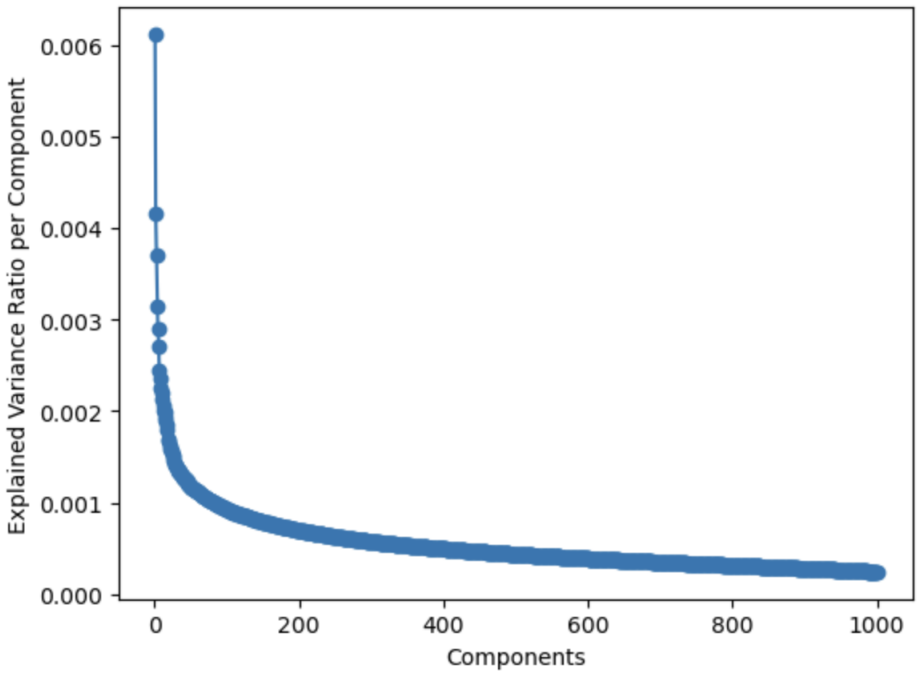


Figure 1: Explained Variance Ratio per Component for $r = 1 \sim 1000$

2.5 Question 5

Based on the below measure scores for r = [1 10, 20, 50, 100, 300], the best choice is r = 10. Results are shown in Figure 2 and Figure 3.

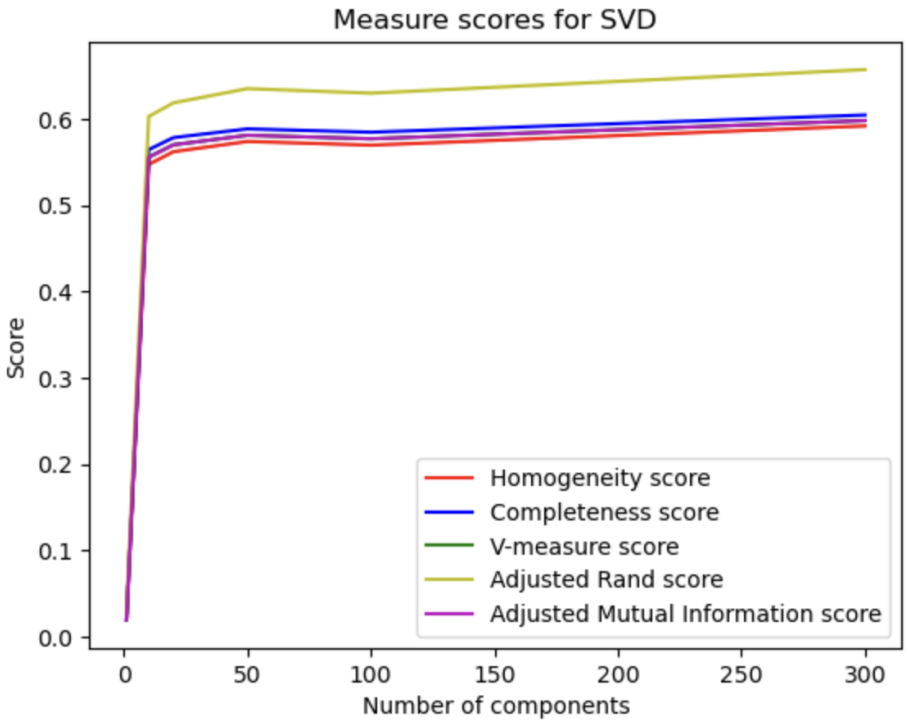


Figure 2: Measure Scores for SVD

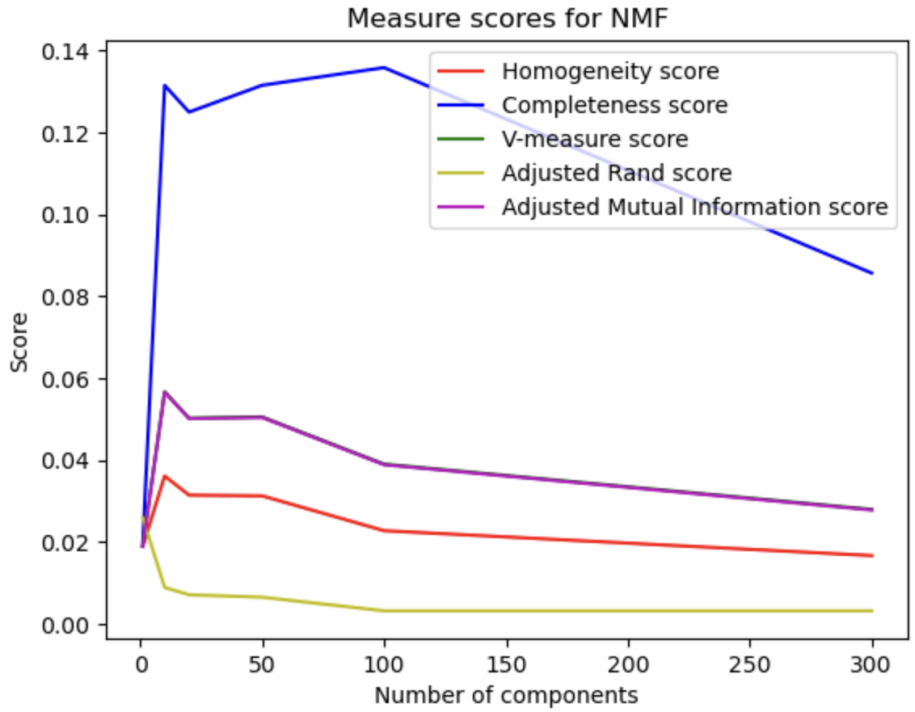


Figure 3: Measure Scores for NMF

2.6 Question 6

Homogeneity measures the extent to which each cluster contains only data points that are members of a single class. As r increases, the homogeneity score may initially improve because smaller clusters may become more homogeneous. However, as r continues to increase, clusters might start to mix different classes, leading to a decrease in homogeneity.

Completeness measures how well all data points belonging to the same class are assigned to the same cluster. Similar to homogeneity, the completeness score may initially increase with r as smaller clusters become more representative of individual classes. However, with a very large r, there's a risk of over-segmentation, causing the completeness to decrease due to splitting classes into multiple clusters.

The V-measure is the harmonic mean of homogeneity and completeness. The non-monotonic behavior arises because, as r increases, homogeneity and completeness may change in opposite directions. Because the V-measure is influenced by the balance between the two, it may have non-monotonic behavior.

The Adjusted Rand Index and Adjusted Mutual Info Scores measure the similarity between the true and predicted clusterings, adjusted for chance. The behavior of this index can be non-monotonic because, with a moderate r, clusters may align well with true classes. However, as ' becomes very large, chance alignments may increase, leading to a decrease in the Adjusted Rand Index and Adjusted Mutual Info Scores.

2.7 Question 7

Compared to Question 3, the scores for SVD are slightly worse and the scores for NMF are significantly worse.

measure	average score
Homogeneity Score	0.47698932483312423
Completeness Score	0.47698932483312423
V-measure	0.48324834320117455
Adjusted Rand Index	0.5279557620030163
Adjusted Mutual Information Score	0.4832004772240264

Table 2: Average Scores for SVD + K-Means.

measure	average score
Homogeneity Score	0.026237508085790017
Completeness Score	0.1047590007332015
V-measure	0.040619403079575504
Adjusted Rand Index	0.00918794478725691
Adjusted Mutual Information Score	0.040485049975417825

Table 3: Average Scores for NMF + K-Means.

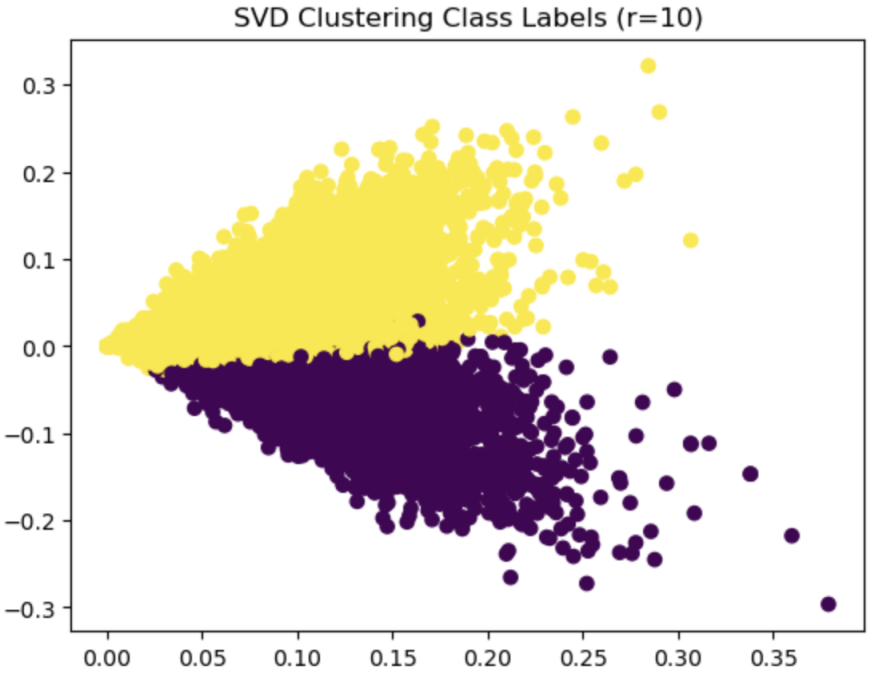


Figure 4: SVD Clustering Results (r = %10)

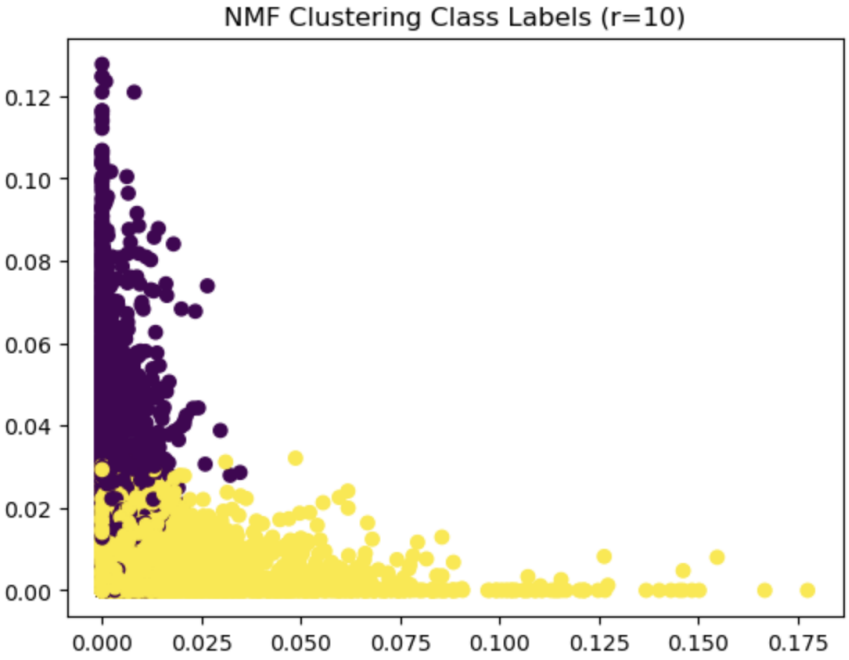


Figure 5: NMF Clustering Results (r = %10)

2.8 Question 8

Figure 4 and Figure 5 (above) are the clustering results for SVD and NMF with the optimal choice of r = 10 for K-Means clustering.

2.9 Question 9

The differences in the distribution of data points between the SVD and NMF K-Means clustering results can be attributed to the underlying constraints in SVD and NMF. Because SVD doesn't enforce non-negativity constraints on the factorized matrices, it captures both positive and negative relationships between features in the data; clusters may be influenced by both positive and negative values. This can be seen in the SVD plot, where the data points of one cluster has only negative y-values. NMF does enforces non-negativity constraints on the factorized matrices, so it instead captures additive relationships between features; the clusters are influenced by positive values only. This can be seen in the NMF plot, where data points belonging to any cluster have only positive x and y-values. The ideal distribution of data for k-means clustering depends on specific characteristics of your data. For example, SVD is a useful clustering method when there is negative and positive data present in the dataset, but NMF is more useful when dealing with data where non-negativity is essential (eg. text data, image data with pixel intensities, etc.)

2.10 Question 10

We use NMF and SVD to reduce the dimensionality of TF-IDF matrix. Before dimensionality reduction, the TF-IDF matrix is very sparse, but after applying these two methods, the reduced matrix becomes dense. To determine the parameters *n_component* for both NMF and SVD, we sweep over the range $\{20, 50, 100, 200\}$, and report the corresponding evaluation metrics in Table 4 and Table 5. The best settings for SVD and NMF are $n\_component = 50$ and $n\_component = 20$, respectively. The contingency matrices of the best SVD and the bset NMF are shown in Figure 6 and Figure 7, respectively.

n_component	20	50	100	200
Homogeneity Score	0.3718	**0.4195**	0.3984	0.4082
Completeness Score	0.3295	**0.3502**	0.3375	0.3160
V-measure	0.3467	**0.3817**	0.3654	0.3563
Adjusted Rand Index	0.1144	**0.1187**	0.1090	0.0991
Adjusted Mutual Information Score	0.3472	**0.3795**	0.3632	0.3539

Table 4: Five clustering metrics of SVD+K-Means with different $n\_component = 20, 50, 100, 200$.

n_component	20	50	100	200
Homogeneity Score	**0.3771**	0.3077	0.2281	0.3165
Completeness Score	**0.3138**	0.1866	0.1140	0.1226
V-measure	**0.3426**	0.2323	0.1521	0.1767
Adjusted Rand Index	**0.0880**	0.0245	0.0102	0.0085
Adjusted Mutual Information Score	**0.3403**	0.2292	0.1484	0.1727

Table 5: Five clustering metrics of NMF+K-Means with different $n\_component = 20, 50, 100, 200$.

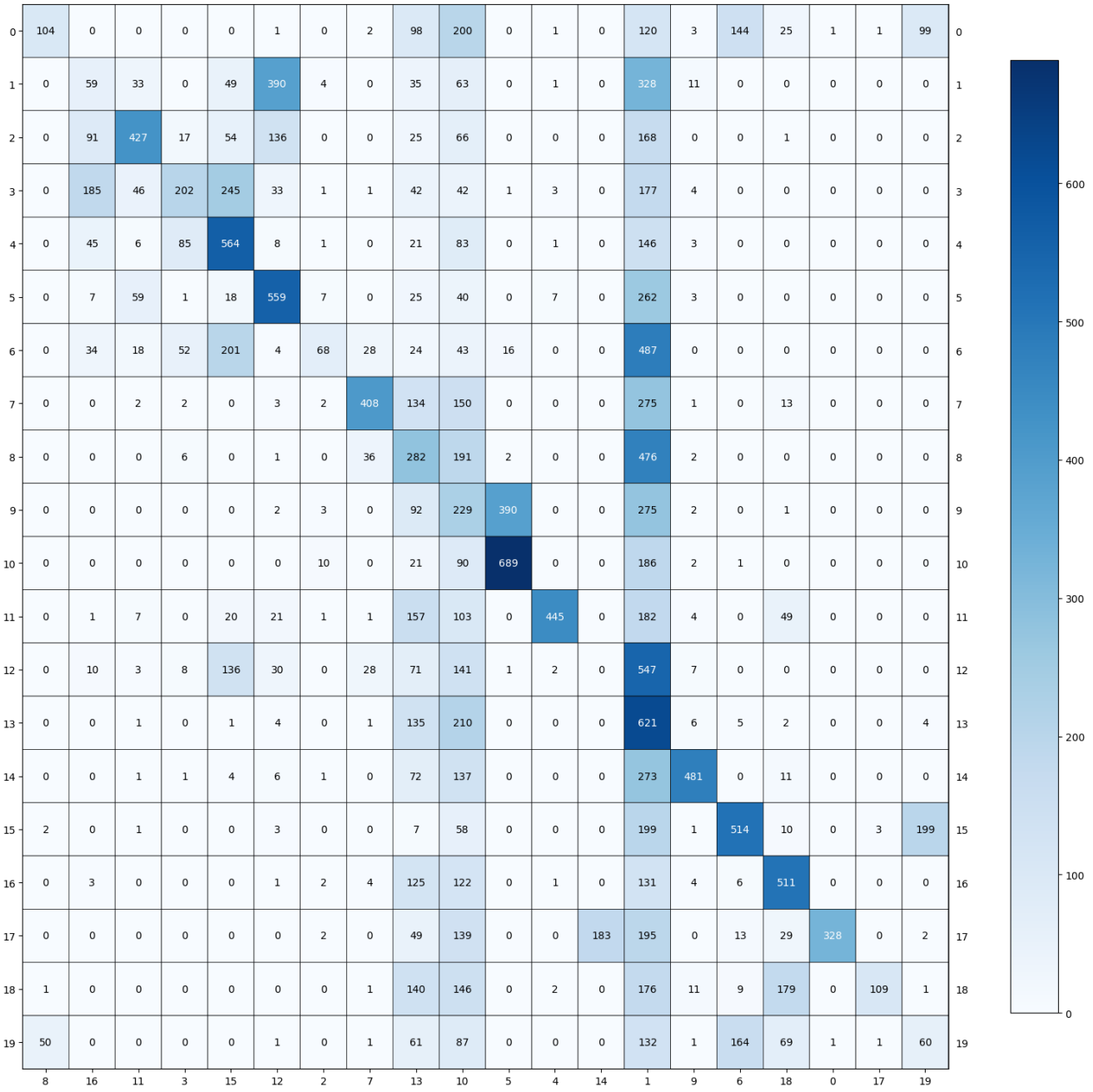


Figure 6: Confusion matrix of SVD+Kmeans with $n\_component = 50$

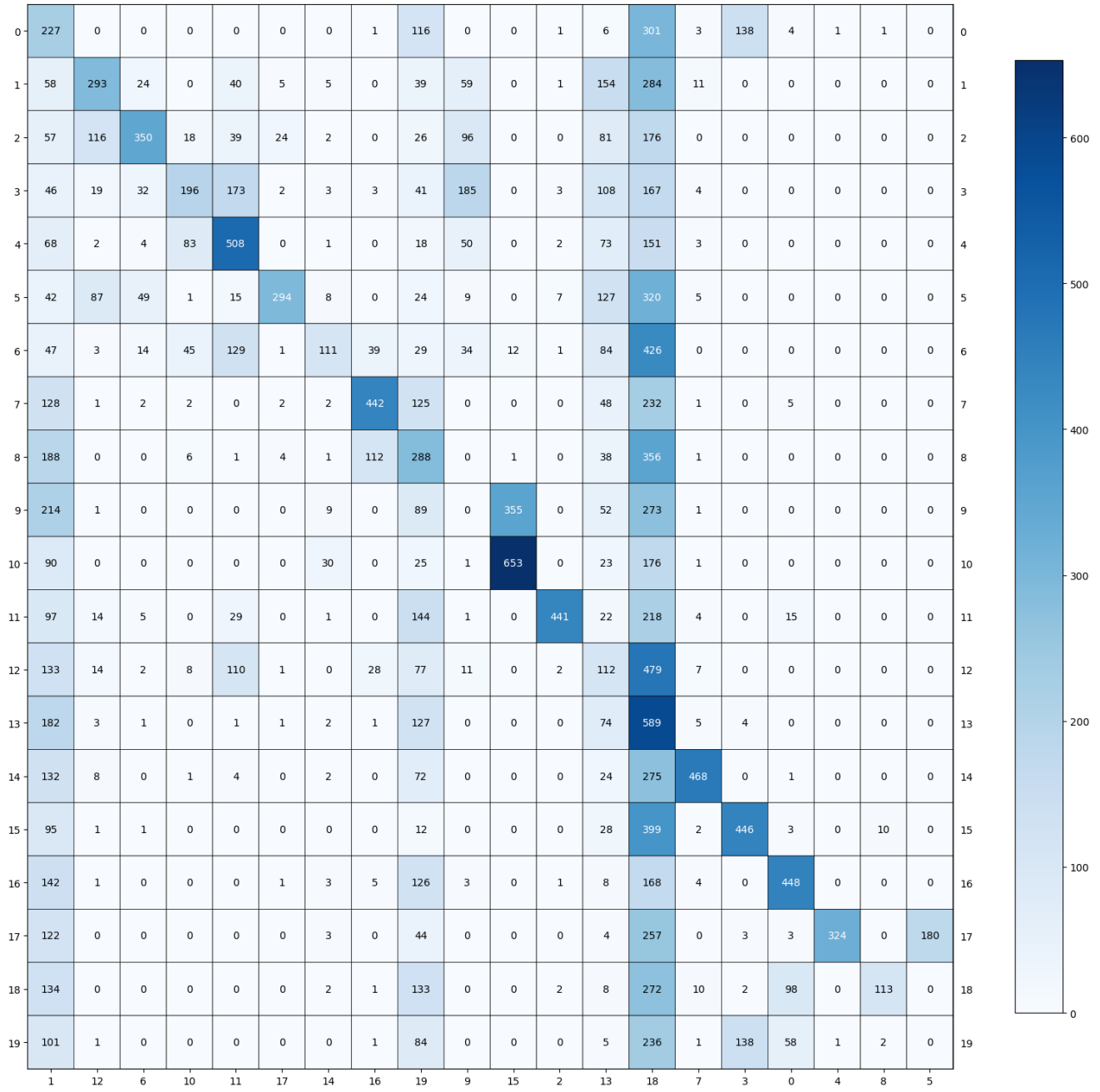


Figure 7: Confusion matrix of NMF+Kmeans with $n\_component = 20$

2.11 Question 11

Five clustering metrics are reported in Table 6 and Table 7. Permuted contingency matrices are shown in Figure 8 $\sim$ Figure 13.

n_component	5	20	200
Homogeneity Score	0.5915	**0.5964**	0.5796
Completeness Score	0.5619	**0.5712**	0.5609
V-measure	0.5763	**0.5835**	0.5701
Adjusted Rand Index	0.4457	**0.4487**	0.4360
Adjusted Mutual Information Score	0.5749	**0.5821**	0.5687

Table 6: Five clustering metrics of UMAP(cosine)+Kmeans with different $n\_component = 5, 20, 200$.

n_component	5	20	200
Homogeneity Score	0.0178	0.0169	**0.0179**
Completeness Score	0.0163	0.0157	**0.0165**
V-measure	0.0170	0.0163	**0.0172**
Adjusted Rand Index	0.0028	**0.0031**	**0.0031**
Adjusted Mutual Information Score	0.0137	0.0130	**0.0139**

Table 7: Five clustering metrics of UMAP(euclidean)+Kmeans with different $n\_component = 5, 20, 200$.

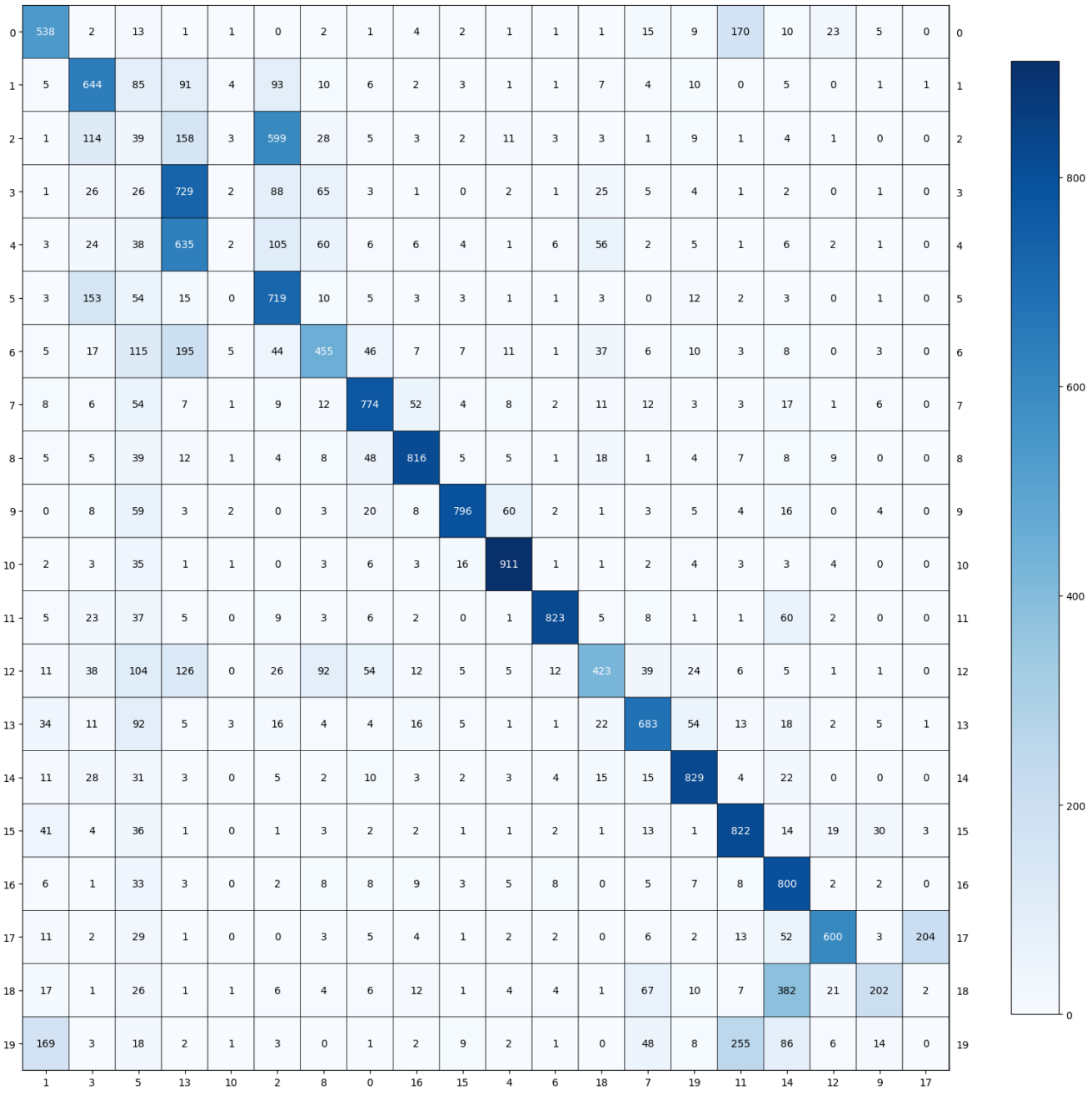


Figure 8: Confusion matrix of UMAP(cosine)+Kmeans with $n\_component = 5$

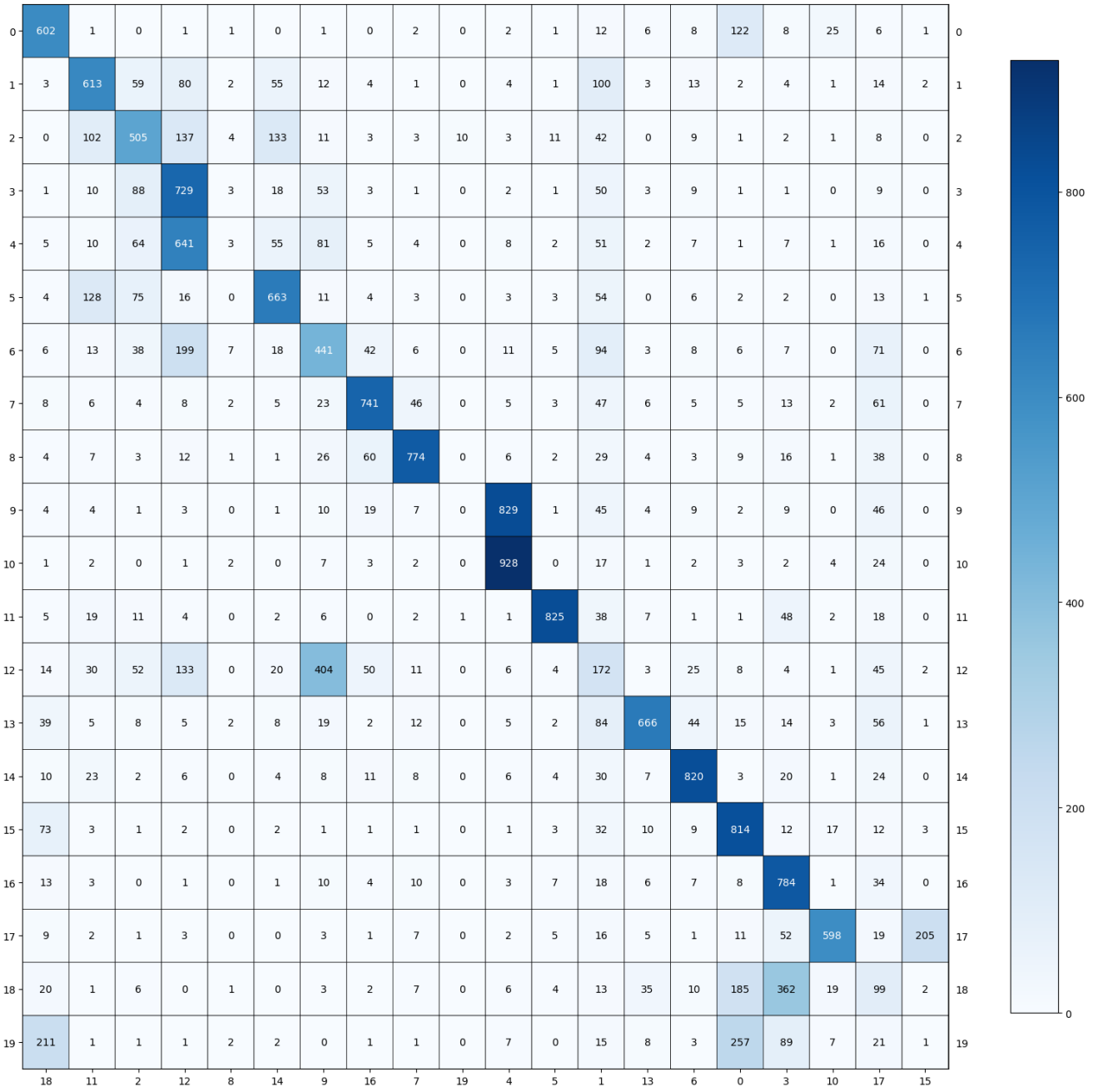


Figure 9: Confusion matrix of UMAP(cosine)+Kmeans with $n\_component = 20$

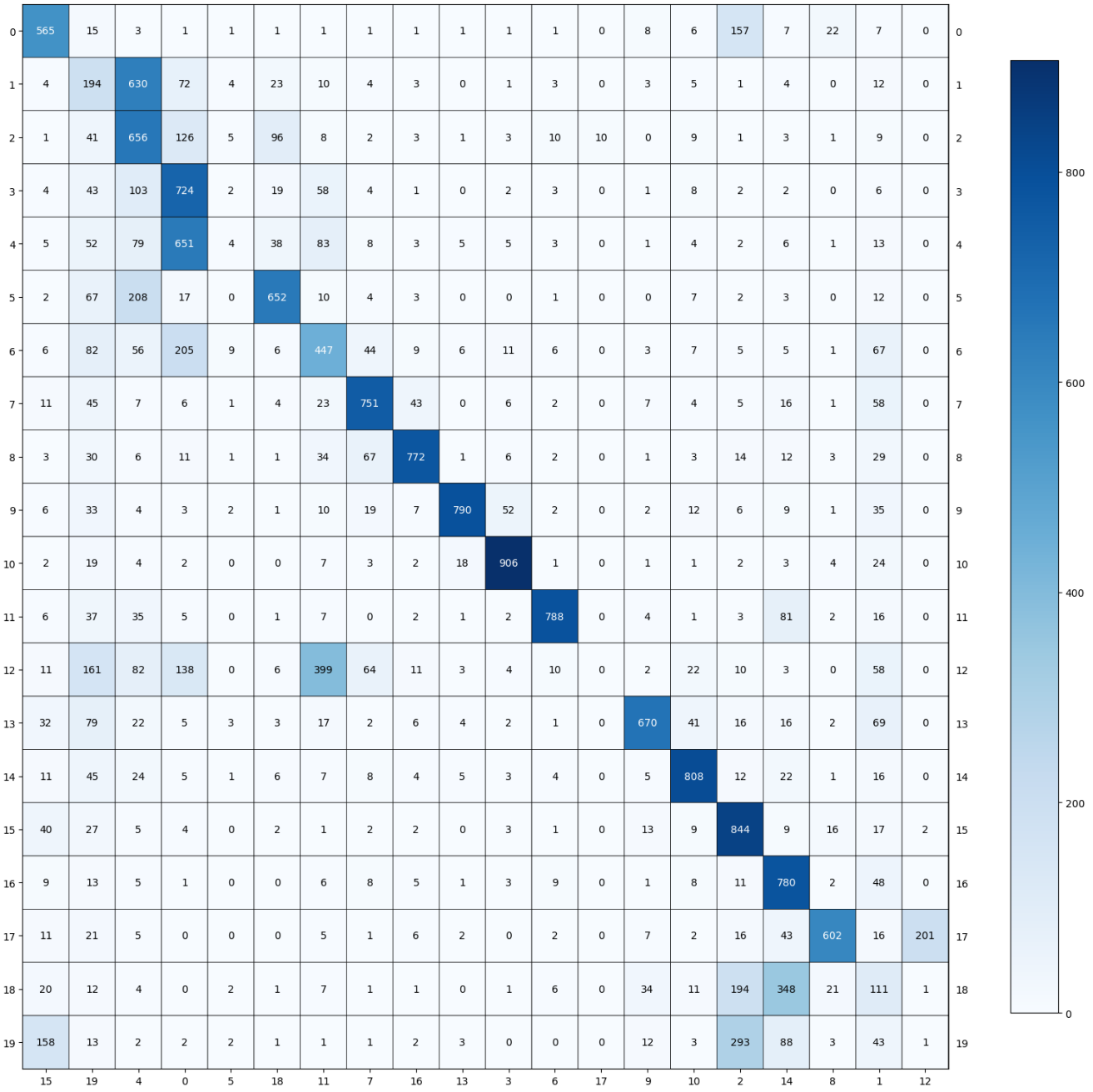


Figure 10: Confusion matrix of UMAP(cosine)+Kmeans with $n\_component = 200$

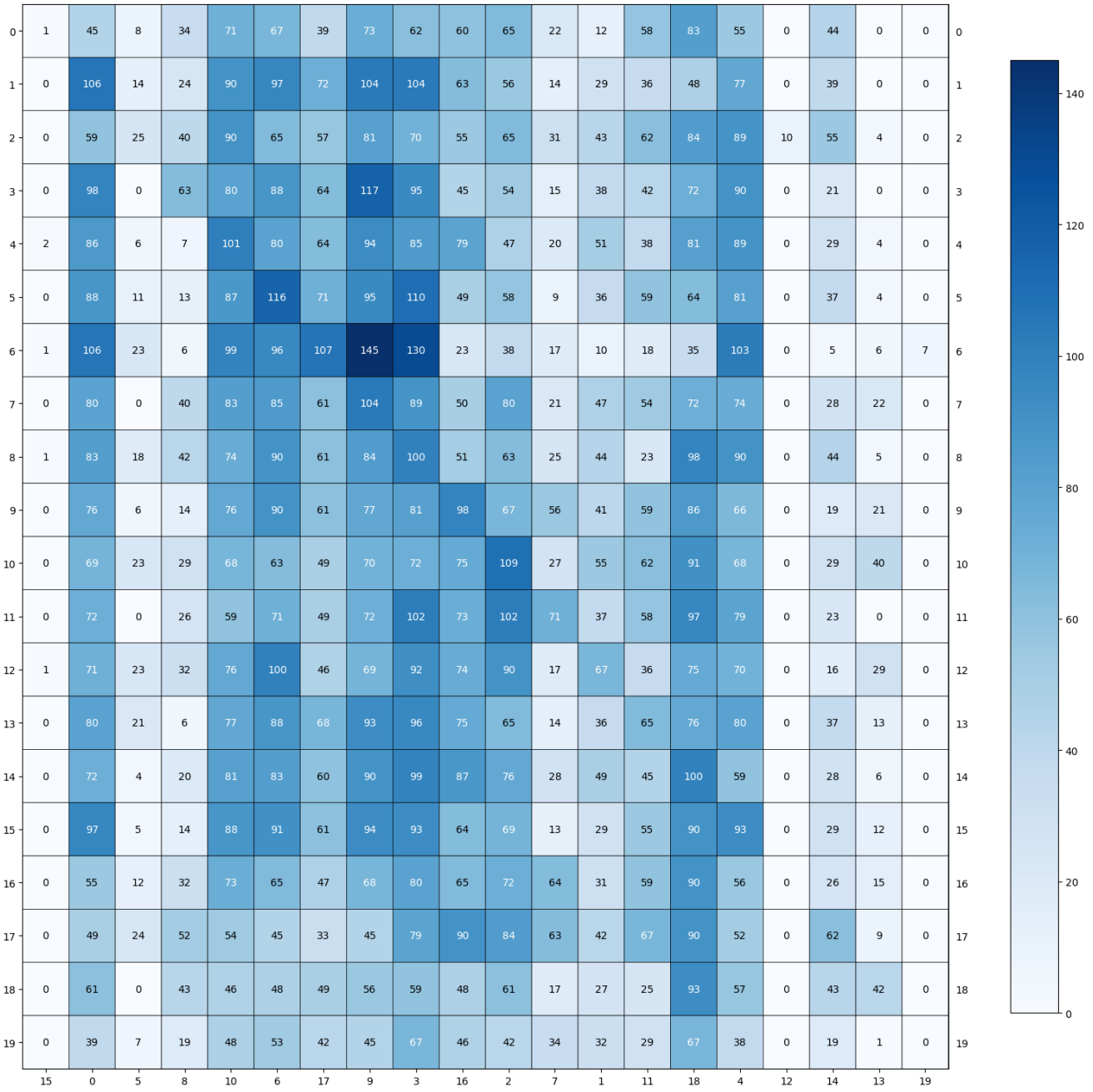


Figure 11: Confusion matrix of UMAP(euclidean)+Kmeans with $n\_component = 5$

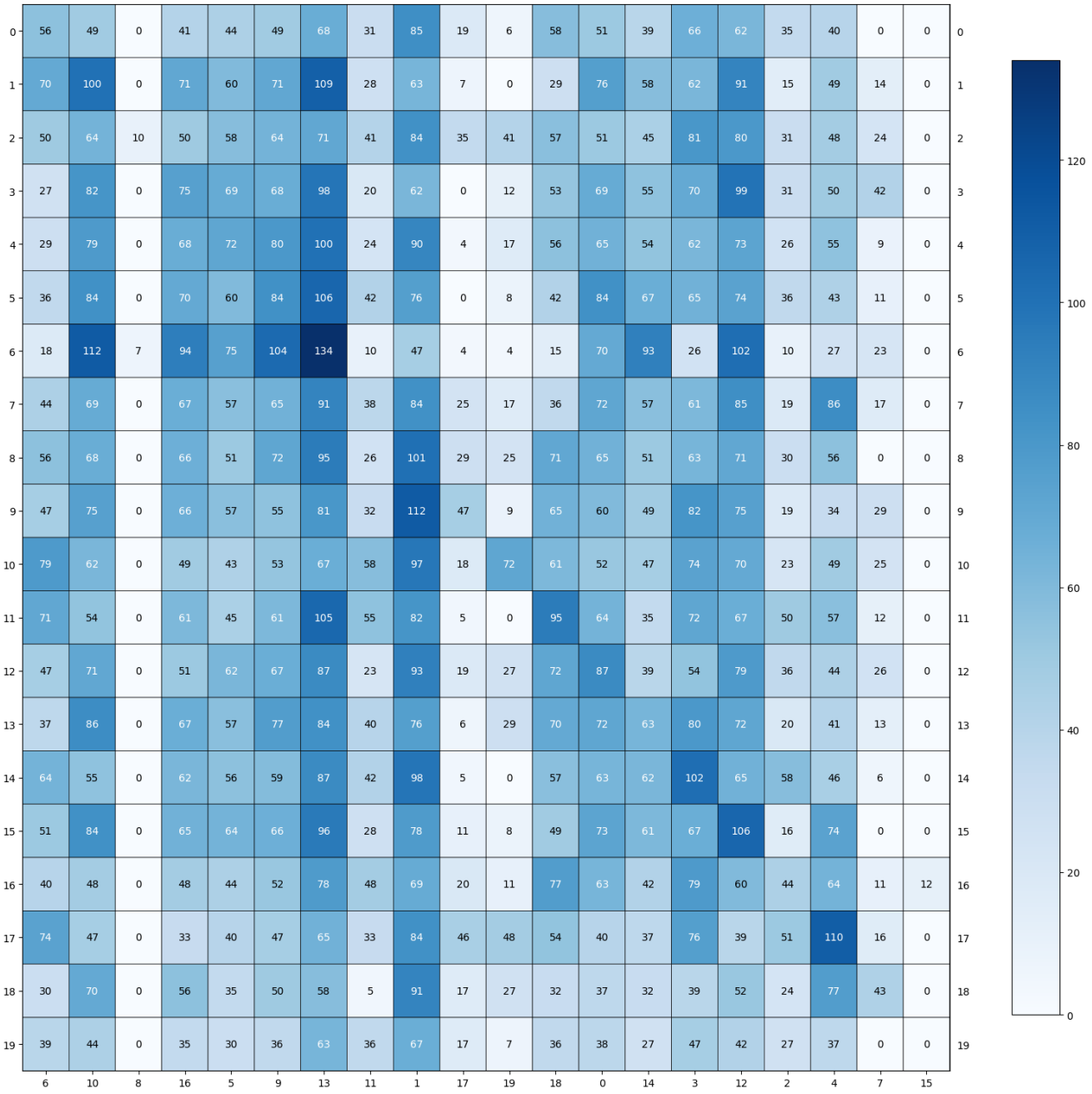


Figure 12: Confusion matrix of UMAP(euclidean)+Kmeans with $n\_component = 20$

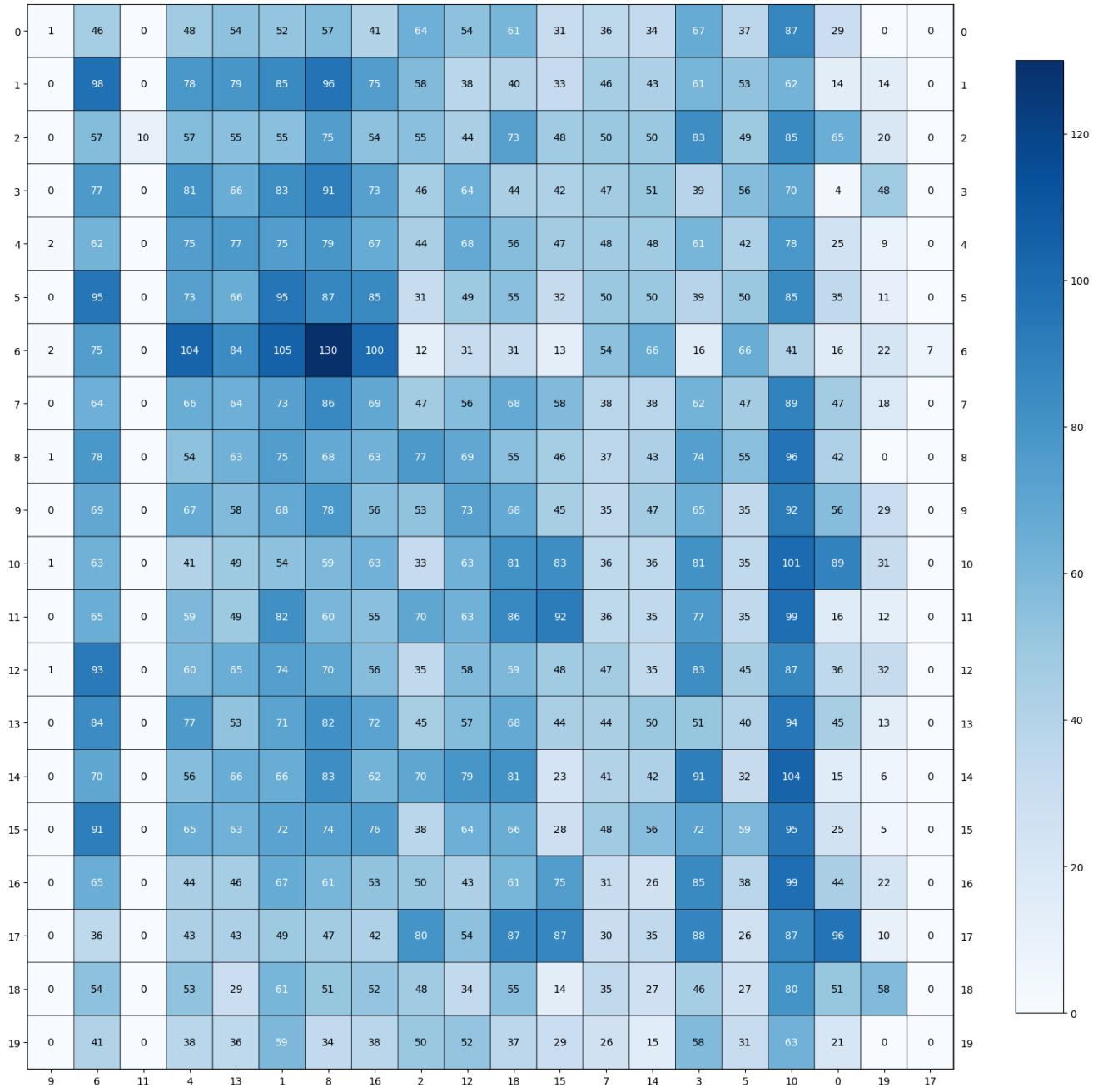


Figure 13: Confusion matrix of UMAP(euclidean)+Kmeans with $n\_component = 200$

2.12 Question 12

From Table 6, we know $n\_component = 20$ is the best for UMAP(cosine)+Kmeans. From Table 7, we know $n\_component = 200$ works the best for UMAP(euclidean)+Kmeans. Analyzing Figure 8 $\sim$ 10, we find that UMAP does not work well with Euclidean metric. Because there is no clear diagonal line in these three contigency matrices. However, there are clear diagnoal lines in Figure 8 $\sim$ Figure 10, indicating UMAP+Kmeans works well with Cosine metric.

2.13 Question 13

We report five evluation metrics of K-Means clustering with 4 different techniques, i.e., TF-IDF representation, PCA-reduced, NMF-reduced reduced, and UMAP-reduced as in Table 8.

Representation	TF-IDF	PCA(SVD)	NMF	UMAP
Homogeneity Score	0.4046	0.4195	0.3771	0.6037
Completeness Score	0.3488	0.3502	0.3138	0.5767
V-measure	0.3746	0.3817	0.3426	0.5899
Adjusted Rand Index	0.1208	0.1187	0.0880	0.4591
Adjusted Mutual Information Score	0.3724	0.3795	0.3403	0.5885

Table 8: Five clustering metrics of K-means with different representation learning techniques, where the reported results of PCA(SVD), NMF, and UMAP are the best chosen from the experiment above.

2.14 Question 14

Results of Agglomerative is reported in Table 9.

	Agglomerative Clustering	
Representation	ward linkage	single linkage
Homogeneity Score	**0.5847**	0.3855
Completeness Score	**0.5510**	0.0166
V-measure	**0.5673**	0.0320
Adjusted Rand Index	**0.4109**	0.0004
Adjusted Mutual Information Score	**0.5659**	0.0269

Table 9: Five clustering metrics of UMAP+K-Means with different linkage, i.e., ward and single.

2.15 Question 15

Five evaluation metrics of HDBSCAN working with UMAP-transformed data is reported in Table 10.

	HDBSCAN		
min_cluster_size	20	100	200
Homogeneity Score	0.4522	0.6118	0.6057
Completeness Score	0.4372	0.4012	0.4085
V-measure	0.4445	0.4846	0.4879
Adjusted Rand Index	0.0847	0.2043	0.2043
Adjusted Mutual Information Score	0.4323	0.4869	0.4869

Table 10: Five clustering metrics of UMAP+HDBSCAN with different *min_cluster_size* in HDBSCAN and fixed parameter $n\_components = 20$ and $metric = cosine$ in UMAP

.

2.16 Question 16

The contingency matrix for the best clustering model in Question 15 is shown in Figure 14.

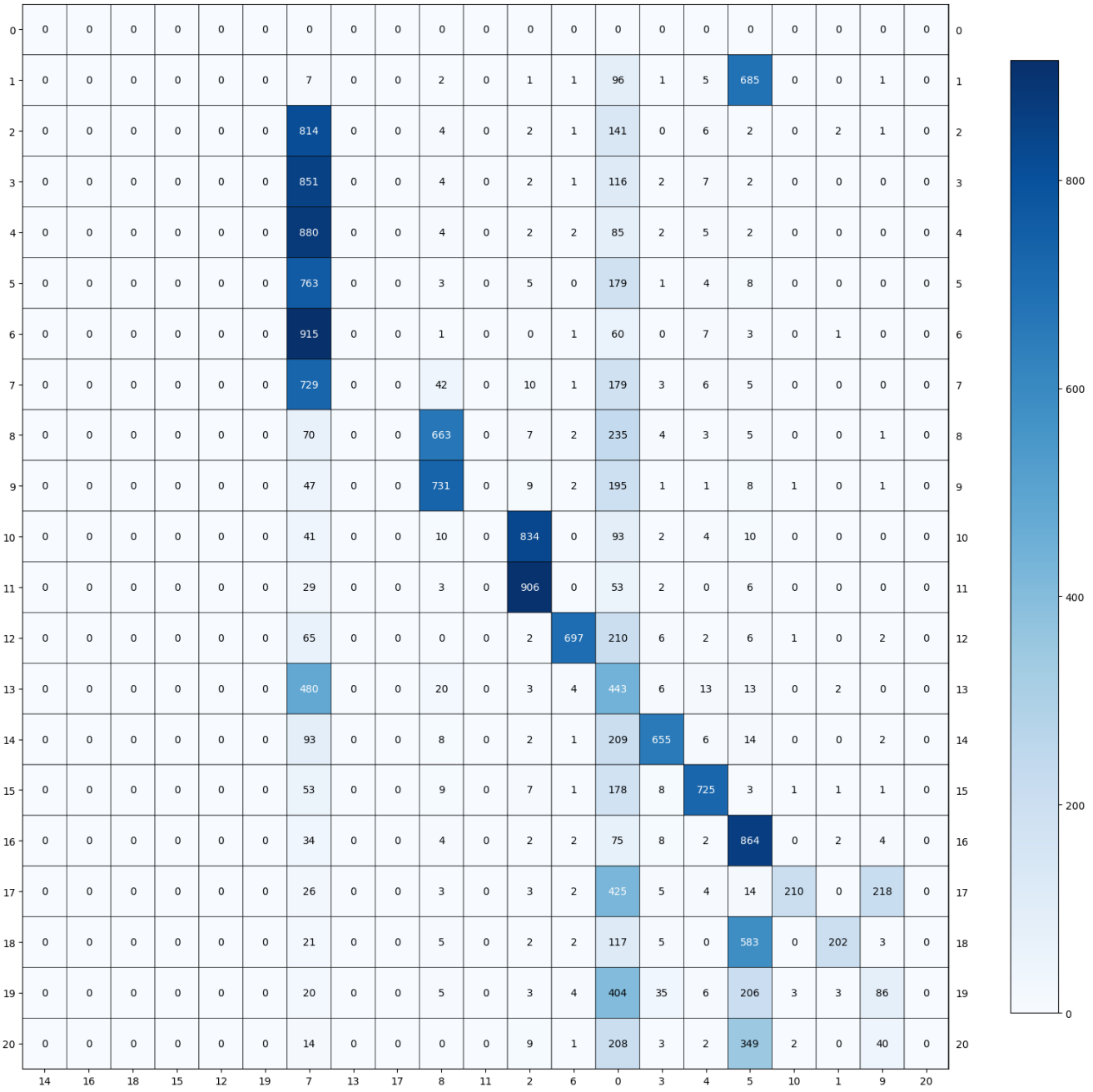


Figure 14: Contingency matrix of UMAP+HDBSCAN with parameters $min\_cluster\_size = 100$ in HDBSCAN and $n\_components = 20$, $metric = cosine$ in UMAP

2.17 Question 17

We sweep all the combinations and parameters and compare adjusted rand scores. Results of different clustering methods with TD-IDF sparse feature are reported in Table 11. Results of different clustering methods with SVD-feature sparse feature are reported in Table 12. Results of different clustering methods with NMF-reduced feature are reported in Table 13. Results of different clustering methods with UMAP-reduced feature are reported in Table 14. Among all tabulated results, Agglomerative Clustering with UAMP (n=5) works the best.

	K-Means			AC	HDBSCAN	
	k=10	k=20	k=50	n=20	min=100	min=200
TF-IDF	0.0966	0.1208	**0.1440**	0.1187	0	0

Table 11: Adjusted rand score of different clustering method working with TF-IDF feature (AC is short for Agglomerative Clustering)

	K-Means			AC	HDBSCAN	
	k=10	k=20	k=50	n=20	min=100	min=200
SVD (n=5)	0.1070	0.1273	0.1009	0.1078	0	0
SVD (n=20)	0.0972	0.1294	0.1406	**0.1669**	0	0
SVD (n=200)	0.0943	0.1106	0.1275	0.1237	0	0

Table 12: Adjusted rand score of different clustering method working with SVD with different parameters $n\_component = 5, 20, 200$ (AC is short for Agglomerative Clustering)

	K-Means			AC	HDBSCAN	
	k=10	k=20	k=50	n=20	min=100	min=200
NMF (n=5)	0.0732	0.0879	0.0777	0.0914	0.0222	0
NMF (n=20)	0.0321	0.0797	0.1254	**0.1420**	0	0
NMF (n=200)	0.0071	0.0114	0.0156	0.01895	0	0

Table 13: Adjusted rand score of different clustering method working with NMF with different parameters $n\_component = 5, 20, 200$ (AC is short for Agglomerative Clustering)

	K-Means			AC	HDBSCAN	
	k=10	k=20	k=50	n=20	min=100	min=200
UMAP (n=5)	0.3488	0.4358	0.3645	**0.4296**	0.2083	0.2083
UMAP (n=20)	0.3327	0.4402	0.3776	0.4076	0.2031	0.2033
UAMP (n=200)	0.3338	0.4425	0.3685	0.3866	0.2053	0.2125

Table 14: Adjusted rand score of different clustering method working with UMAP with different parameters $n\_component = 5, 20, 200$ (AC is short for Agglomerative Clustering)

3 Part 2 - Deep Learning and Clustering of Image Data

3.1 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?

Answers: VGG exhibits key characteristics, including a deep architecture, utilization of small-sized convolutional kernels, and a combination of convolutional, pooling, and fully connected layers. This structural design empowers the network to grasp intricate features within images. The adaptability of VGG makes it particularly effective for feature extraction across diverse datasets, aligning seamlessly with the principles of transfer learning. Consequently, the features derived from VGG networks prove advantageous for feature extraction and discrimination in other datasets.

3.2 Question 20

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

Answers:

The feature extraction is done by

1) Extract VGG-16's feature layers.

2) Encapsulate the extracted list of convolutional layers into 'nn.Sequential' module, faciliating their use in the forward pass of the model.

3) Extract the average pooling layer from the VGG-16 model to reduce the dimensionality of the feature maps.

4) Flatten the feature maps into a one-dimensional vector.

5) Extract the first layer from the fully connected layer of the VGG-16 model to be used as a linear layer for the final classification.

3.3 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?

Answers:

1) There are 3670 images in the dataset. The original images have being composed to the size of 224x224x3=150,528 pixels.

2) According to the output shape of the x, it could be seen that the features:

   * Dimension: 512x7x7 features for the execution of 'features', and 'pooling'
   * Dimension: 25,088 features for the execution of 'flatten'
   * Dimension: 4,096 features for the 'fc'

3.4 Question 22

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

Answers: The extracted features dense. In theory, the TF-IDF features are typically sparse. In this extracted features, the dimension are being reduced from 150,528(224x224x3) to 25,088(512x7x7), and finally to 4,096. The dimensionality reduction operations usually do not lead to sparse matrices; rather, they may decrease the sparsity of the original data. According to the output of the 'out' from the code, from Figure , as shown in Figure 15 we can observe that almost no zero values are shown in the feature vectors.

```
torch.Size([64, 3, 224, 224])
torch.Size([64, 512, 7, 7])
torch.Size([64, 512, 7, 7])
torch.Size([64, 25088])
torch.Size([64, 4096])
 93%|█████████ | 54/58 [00:35<00:02,  1.56it/s]
 The number of zero value is: 0

torch.Size([64, 3, 224, 224])
torch.Size([64, 512, 7, 7])
torch.Size([64, 512, 7, 7])
torch.Size([64, 25088])
torch.Size([64, 4096])
 95%|█████████ | 55/58 [00:35<00:01,  1.60it/s]
 The number of zero value is: 0

torch.Size([64, 3, 224, 224])
torch.Size([64, 512, 7, 7])
torch.Size([64, 512, 7, 7])
torch.Size([64, 25088])
torch.Size([64, 4096])
 97%|█████████ | 56/58 [00:36<00:01,  1.61it/s]
 The number of zero value is: 0

torch.Size([64, 3, 224, 224])
torch.Size([64, 512, 7, 7])
torch.Size([64, 512, 7, 7])
torch.Size([64, 25088])
torch.Size([64, 4096])
 98%|█████████ | 57/58 [00:37<00:00,  1.62it/s]
 The number of zero value is: 0

torch.Size([22, 3, 224, 224])
torch.Size([22, 512, 7, 7])
torch.Size([22, 512, 7, 7])
```

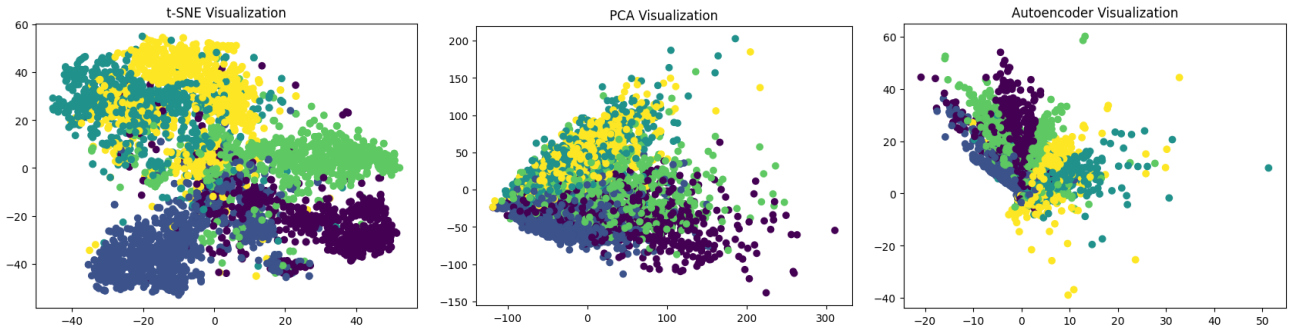
Figure 15: Fig: The number of zeros in the feature vectors

3.5 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.

Answers: T-SNE, PCA, and the Autoencoder have been used to reduce the dimensionality from 4096 dimensions to 2 dimensions. From Figure 16 Notably, the t-SNE visualization reveals distinct clusters, effectively representing five clear groups. In contrast, the PCA visualization faces challenges in clearly distinguishing between these groups, highlighting the dataset's inherent nonlinearity. The Autoencoder also exhibits good separation of different groups in 2D visualization. However, an unexpected clustering phenomenon occurs where one cluster is encapsulated within another, indicating potential complexities in the underlying data structure.

Evaluating the trade-offs between these methods is crucial. While t-SNE excels in capturing intricate cluster patterns, PCA struggles with the dataset's nonlinearity. The Autoencoder, though generally effective, needs careful examination and potential adjustments to address the observed encapsulation issue.



t-SNE visualization PCA visualization Autoencoder visualization

Figure 16: Visualization comparisons of different dimensionality reduction methods

3.6 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.

Answers:

According to the conservative parameter grid over min_cluster_size and min_samples, the best parameters are 5 and 1 respectively (Figure 17). Among the modules performance for different dimensionality reduction and different clustering shown in the table 15. The best results in terms of rand score is: 0.7979427013129371 by UMAP of dimensionality reduction and K-Means of clustering.

HDBSCAN with grid over min_cluster_size & min_samples

```
[ ]  clusterer = hdbscan.HDBSCAN()

     param_grid = {'min_cluster_size': [2, 3, 5, 10],
                   'min_samples': [1,2,5]}

     score = make_scorer(rand_score)
     grid_search = GridSearchCV(clusterer, param_grid, cv=5, scoring=score)

     grid_search.fit(umap_features)
```

```
[ ]  best_params

     {'min_cluster_size': 5, 'min_samples': 1}
```

Figure 17: The grid searching for parameters of HDBSCAN

Dimensionality Reduction Clustering	Nonen (N/A)	SVD (r=50)	UMAP (n_components=50)	Autoencoder (num_features=50)
K-Means (k=5)	0.703	0.723	0.798	0.714
Agglomerative Clustering (n_clusters=5)	0.710	0.670	0.788	0.701
HDBSCAN (min_cluster_size=5) (min_samples=1)	0.710	0.461	0.759	0.254

Table 15: The performance results of different dimensionality reduction and clustering

3.7 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.

Answers: According to Question 24, the dimensionality reduction algorithm will be UMAP with n_components=50. Certainly, the model's performance experiences a mild decline with the reduced-dimension representations, although it is not deemed significant. The success in classification is intuitive, given the supervised nature of the learning, whereas clustering relies on unsupervised methods. From the Figure 18, notably, as the dimensionality reduces from the original 4096 to 50 and then to 2, the Rand accuracy decreases from 93.7% to 88.4% and finally to 86.6%. This reduction might be attributed to the loss of intricate features during the dimensionality reduction process.

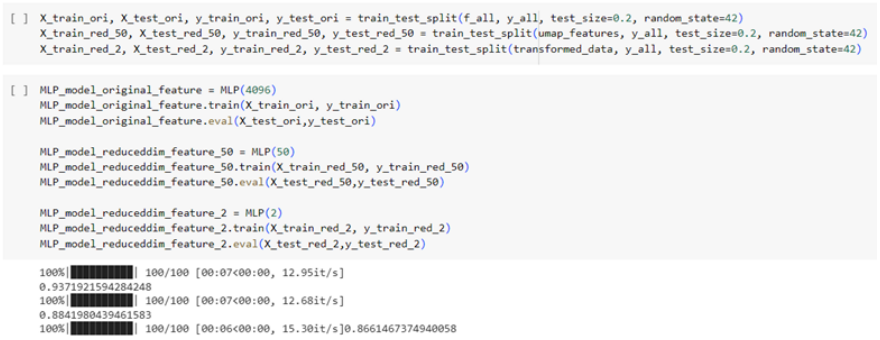


Figure 18: The test accuracy of the MLP classifier

4 Part 3 - Clustering using both image and text

4.1 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.

Answers: For the Bug (Figure 19), Fire (Figure 20), and Grass types (Figure 21, and Figure 22), finding corresponding images is straightforward as these types have distinct and recognizable visual representations. It's interesting to note that when querying for "Grass" alone (Figure 21), it displays two different types, namely Psychic and Rock. Upon closer inspection, these Pokémon share a green color and a visual resemblance to grass. When refining the query to "Type1: Grass" (Figure 22) the search exclusively shows Pokémon with Grass type as their primary classification. However, challenges arise for Dark (Figure 23) and Dragon types (Figure 25). Dark may be interpreted as a descriptor for color, making it less visually distinguishable, while Dragon might denote an overall appearance characteristic rather than a specific visual trait. Even added the "type" in front of the dark (Figure 24)and dragon (Figure 26), the performance aren't much improved. Hence, the difficulty in locating images for Dark and Dragon types can be attributed to the abstract or subjective nature of these descriptors.

Figure 19: Top five search results when query is: Bug

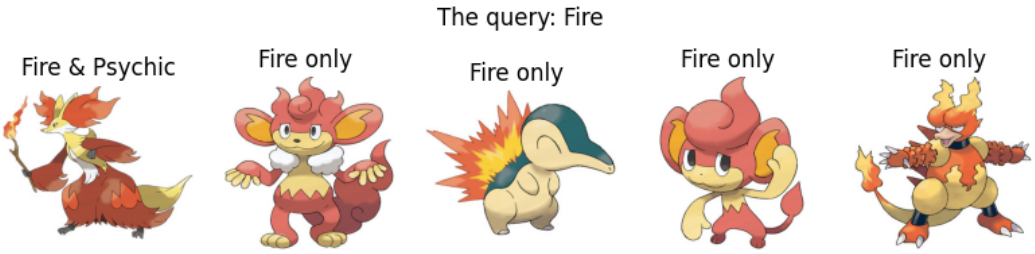


Figure 20: Top five search results when query is: Fire

Figure 21: Top five search results when query is: Grass

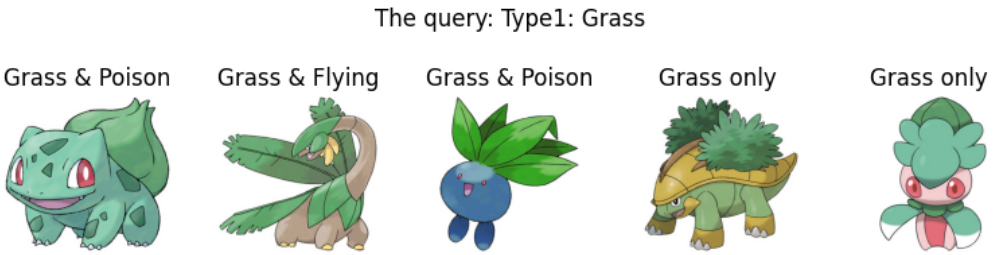


Figure 22: Top five search results when query is: Type1: Grass

Figure 23: Top five search results when query is: Dark

Figure 24: Top five search results when query is: Type1: Dark

Figure 25: Top five search results when query is: Dragon

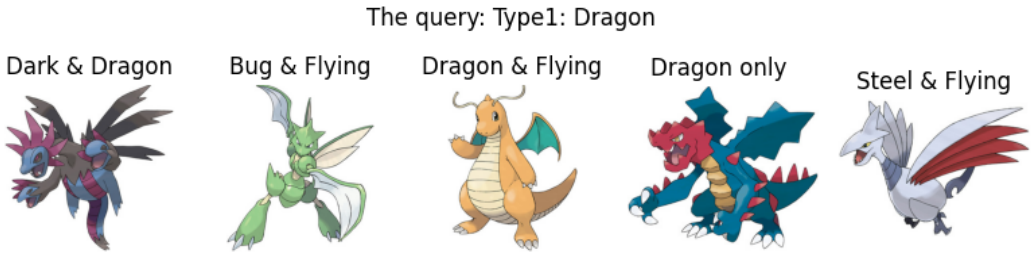


Figure 26: Top five search results when query is: Type1: Dragon

4.2 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.

Answers: All the inferences are based on the images, and as a result shown in Figure 27, types directly related to visual appearance show higher accuracy and more confident predictions. For example, Umbreon, being a Dark-type, has a prediction accuracy of over 50%, which aligns with the dark color of its appearance. Stakataka, with its blocky rock-like structure, achieves nearly 100% accuracy in the Rock type prediction. Torkoal presents an interesting case where the gases it emits are mispredicted as Ice, likely due to their white color, while its shell is identified as Rock. For Pokémon with appearance characteristics that don't strongly indicate a specific type, like Hattrem, the Psychic type becomes more challenging to determine. In summary, the model tends to perform better when visual characteristics strongly correlate with specific Pokémon types.

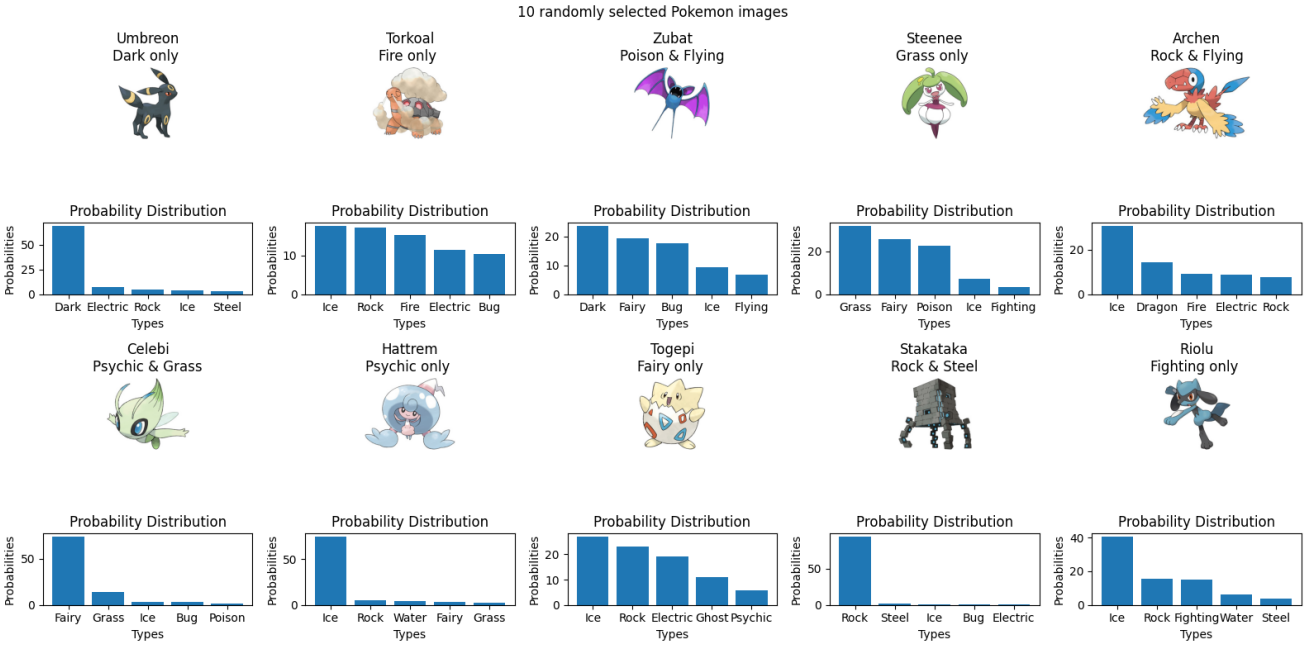


Figure 27: 10 randomly selected Pokemon images & their predicted types

4.3 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.

Answers:

As shown in Figure 28, the clustering of Pokemon types makes sense, as there is a clear division between Grass (blue), Fire (red), and Bug (green). The Grass type is located at the left bottom corner of the 2D visualization plot, while the Fire type is at the left top, and the Bug type is at the right bottom. There are also cases where the color represents the second type of the Pokemon. Additionally, there are Pokemon with mixed types, contributing to the overall diversity. In general, the division aligns with the known relationships between different Pokemon types.

Figure 28: 2D visualization for types of Bug, Grass, and Fire