ECE-219

Data Representation and Clustering

Team Member Names:

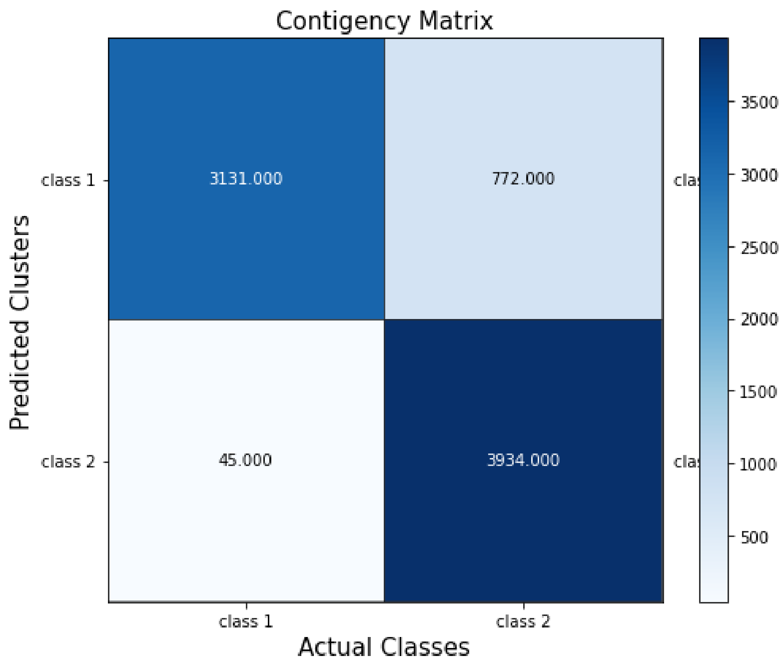
Tianpei Gu, 405863048
Yuxin Huang, 105711853
Yilin Xie, 405729012

Question 1

- The dimensions of the TF-IDF matrix obtained = **(7,882, 23,522)**

Question 2

- The contingency matrix should be squared.



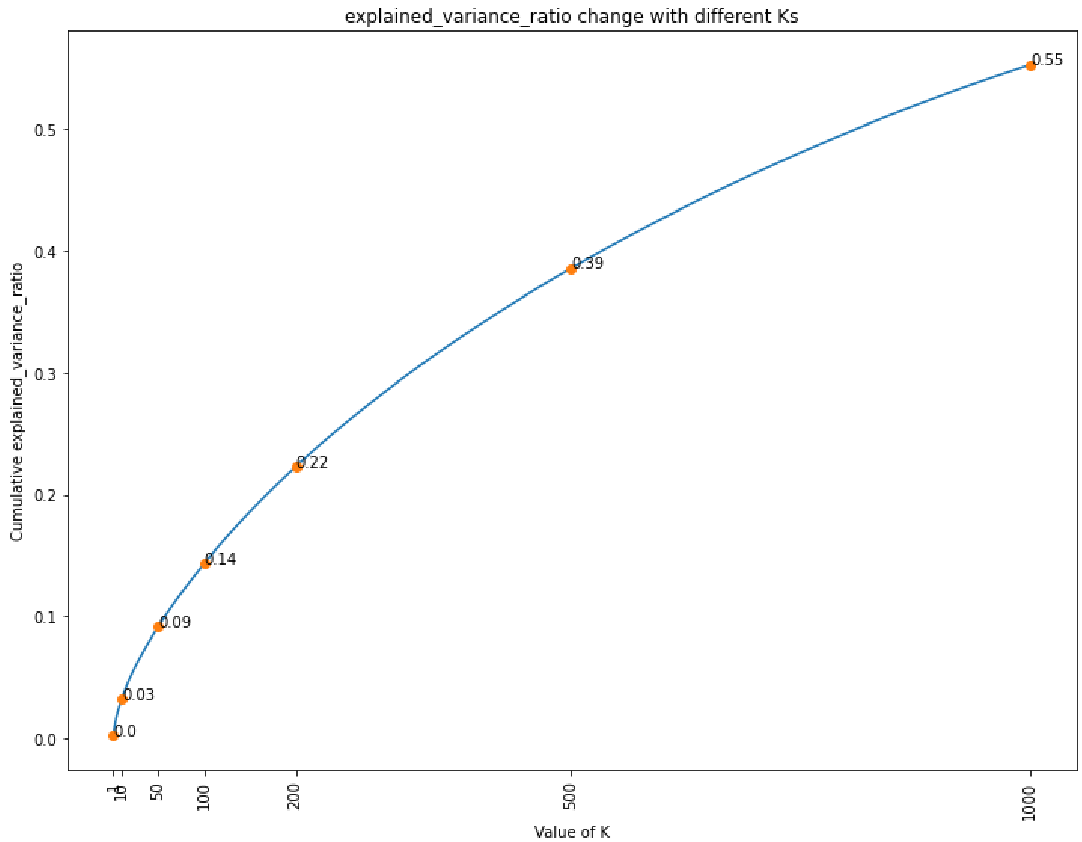
-

Question 3

- `Homogeneity:           0.572286669402203`
- `Completeness:          0.5883415696007297`
- `V-measure:             0.580203076269998`
- `Adjusted Rand:         0.6283142568971879`
- `Adjusted mutual info:  0.5801641116732054`

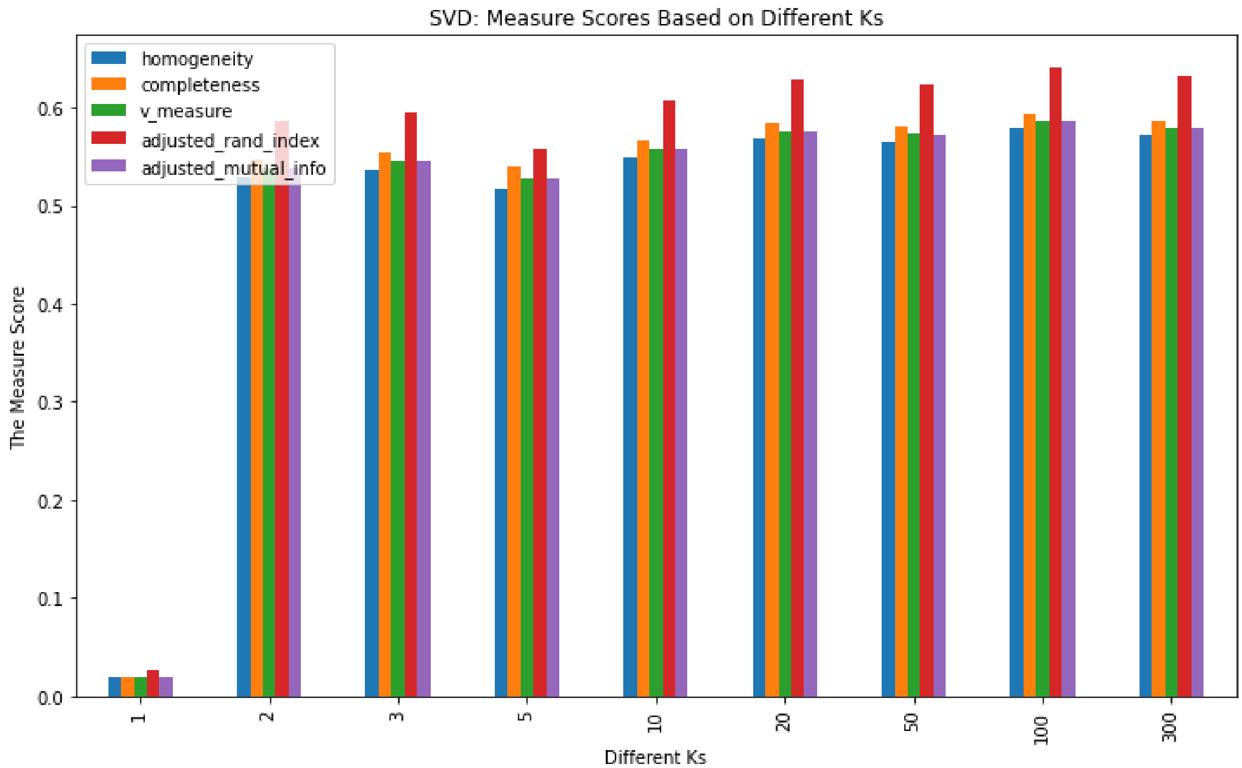
Question 4

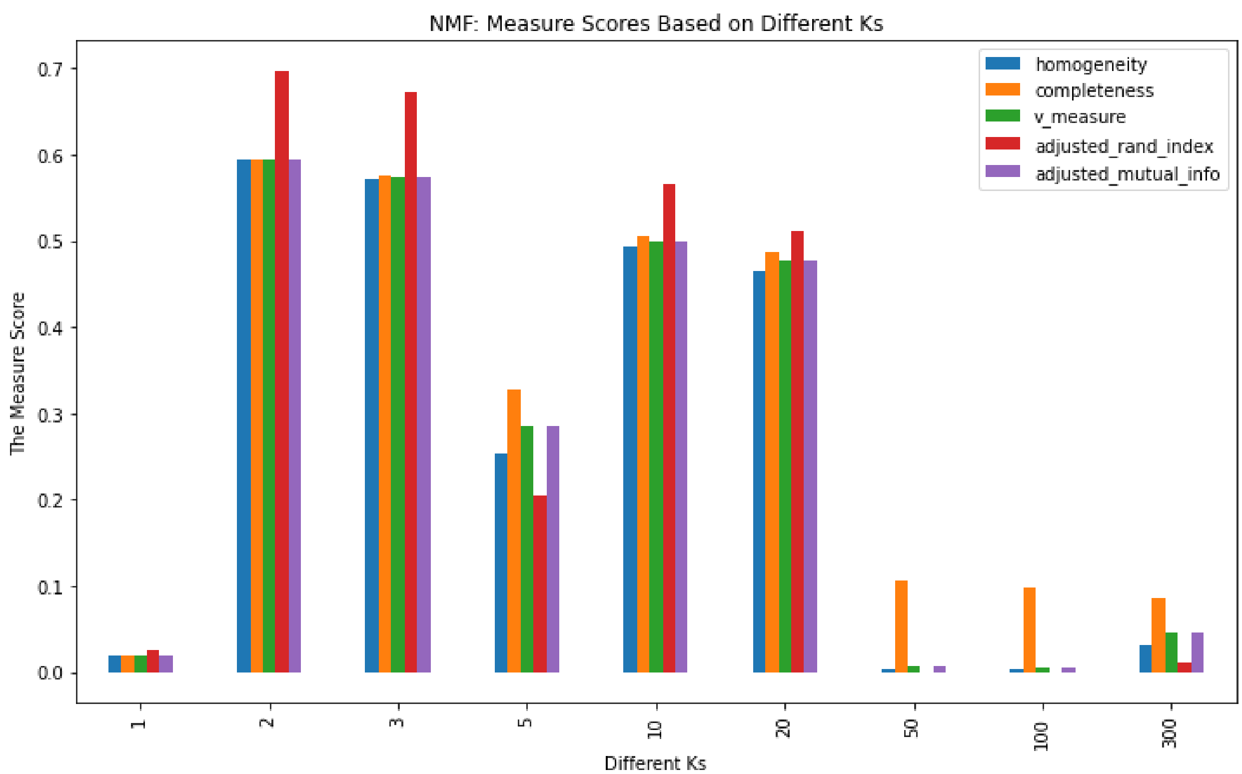
- Percentage of Variance Plot from r = 1 to 1,000



-

Question 5





- Based on the measure score comparison based on different Ks above, the best choice of r for SVD is 100 and that for NMF is 2.

Question 6

- Based on the results in question 5, we can see that as r increases, the performance of K-Means clustering gets better for SVD but as r gets too large the performance gets worse a little bit. The same pattern is also observed in NMF but a little bit broken.

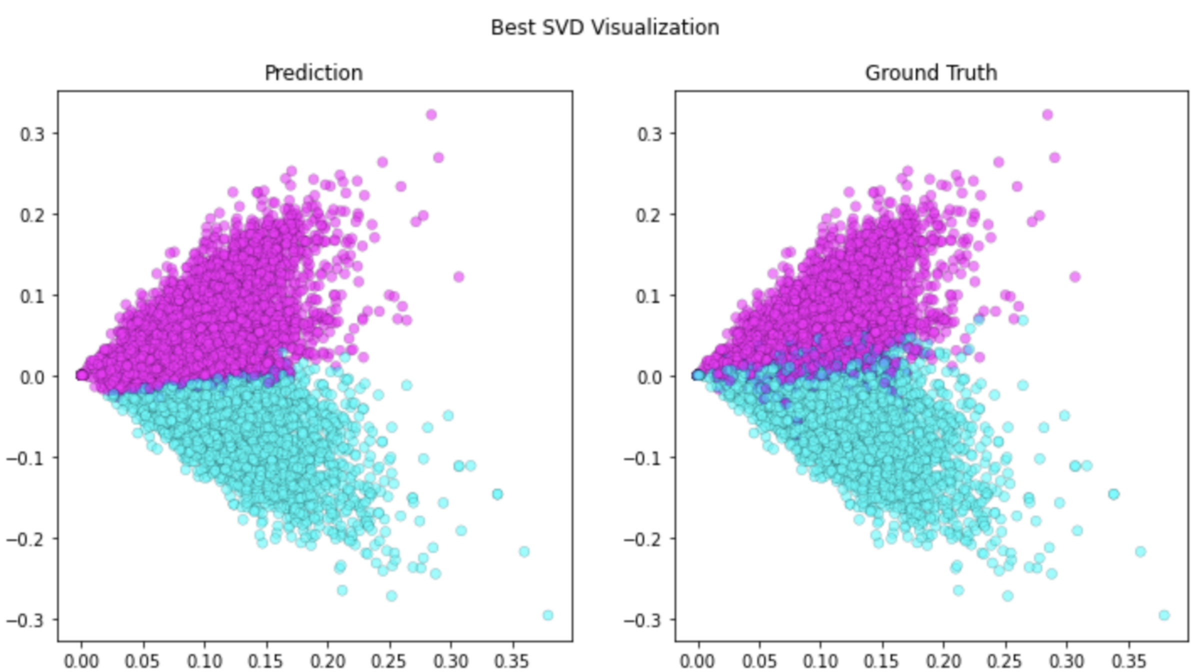
- This is mainly because the greater the number of principal components is, the higher dimensions that K-Means needs to perform clustering will be. As the instruction mentioned in section "**Clustering with Dense Text Representation**", the Euclidean distance is not a good metric for higher-dimensional space because the distances between data points tends to be almost the same. In other words, the marginal information gained per number of principal components is decreasing. Therefore, this explains the non-monotonic behavior in measure scores as r increases.

Question 7

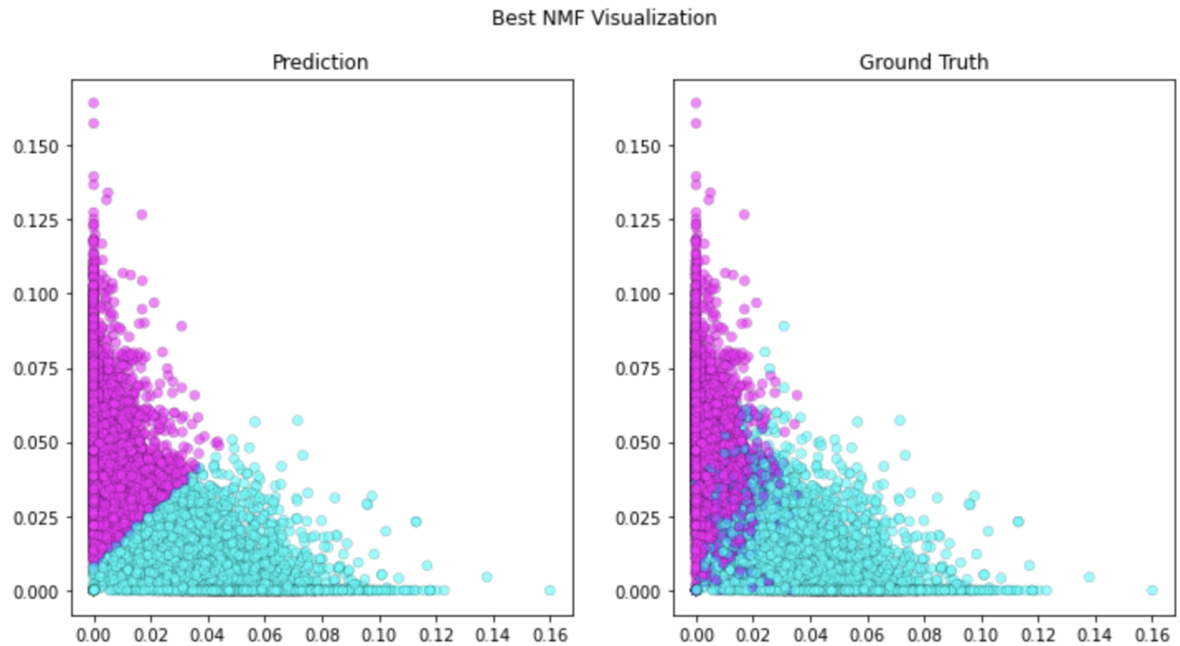
- No, the measure score in question 3 is better than the average score of SVD or NMF.

Question 8

- SVD : r = 100



- NMF r = 2



Question 9

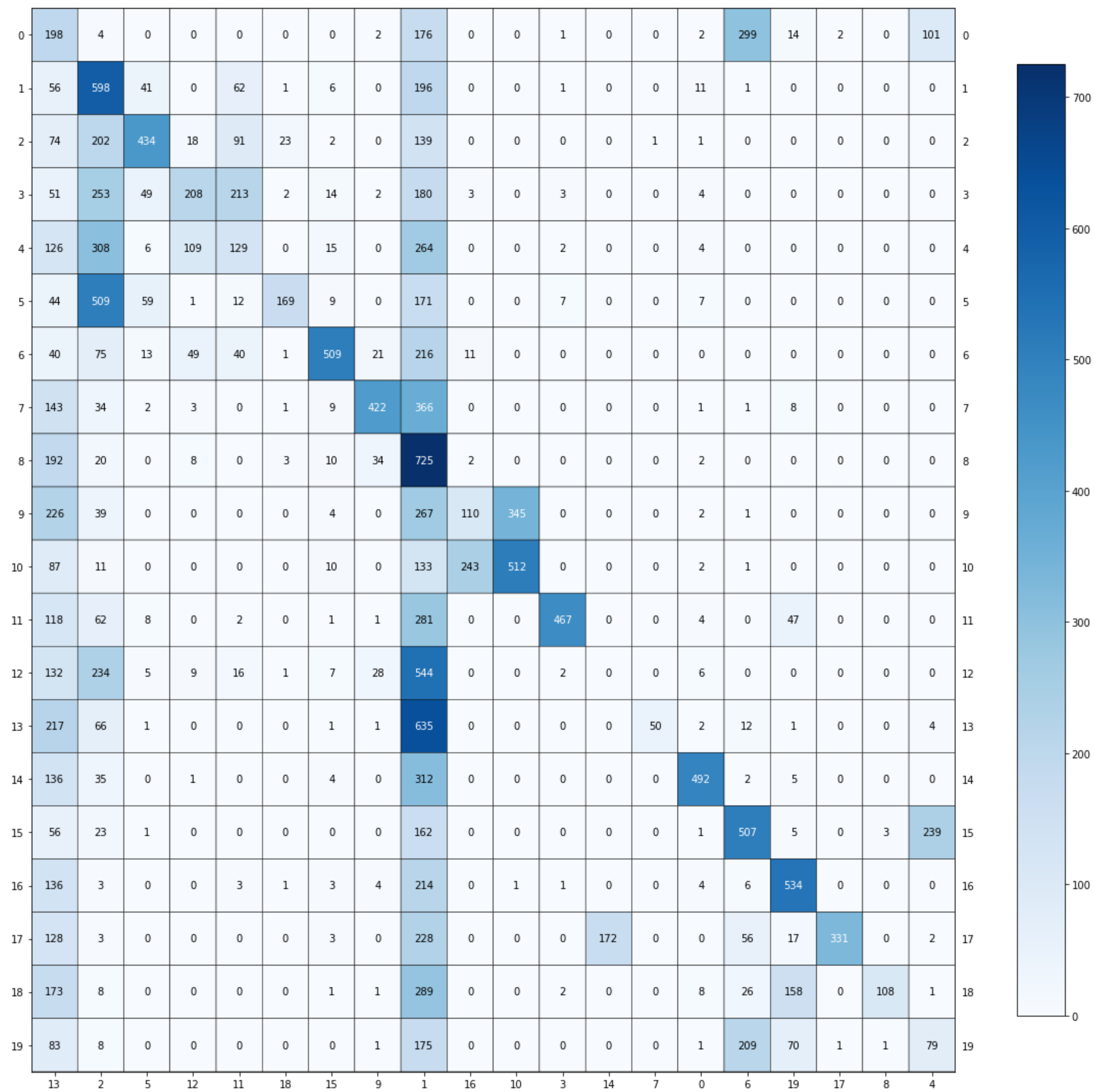
- Comparing the K-Means clustering results with ground truth labels, the clustering result with either SVD or NMF reaches reasonable performance. The datapoints of two classes are approximately separated by the border line. It is comparatively ideal compared to sparse data.

Question 10

- Dimension Reduction used: Compared with NMF, we observe SVD is faster and more accurate. Hence we choose to use **SVD with n_components = 100** according to the five clustering evaluation metrics.
- SVD Metrics Report
    - `==== n_components: 5 ====`
    - `Homogeneity:            0.32467718611007096`
    - `Completeness:           0.35279326312000675`
    - `V-measure:              0.33815179415890084`
    - `Adjusted Rand:          0.12839451940791383`
    - `Adjusted mutual info:   0.3359186292650956`
    - `==== n_components: 20 ====`
    - `Homogeneity:            0.32203003750180664`
    - `Completeness:           0.36925763276468926`
    - `V-measure:              0.3440305807311734`
    - `Adjusted Rand:          0.10456513767905536`
    - `Adjusted mutual info:   0.34175927413275015`
    - `==== n_components: 100 ====`
    - `Homogeneity:            0.35213081962046117`
    - `Completeness:           0.4334267387430322`

- V-measure: 0.3885721959749402
- Adjusted Rand: 0.11948852937487153
- Adjusted mutual info: 0.3863690421217935
- ==== n_components: 200 ====
- Homogeneity: 0.31632490981493006
- Completeness: 0.4074222375789923
- V-measure: 0.35614040904433697
- Adjusted Rand: 0.09276539932106924
- Adjusted mutual info: 0.3537741974913982

- Contingency Matrix (n_components = 100)

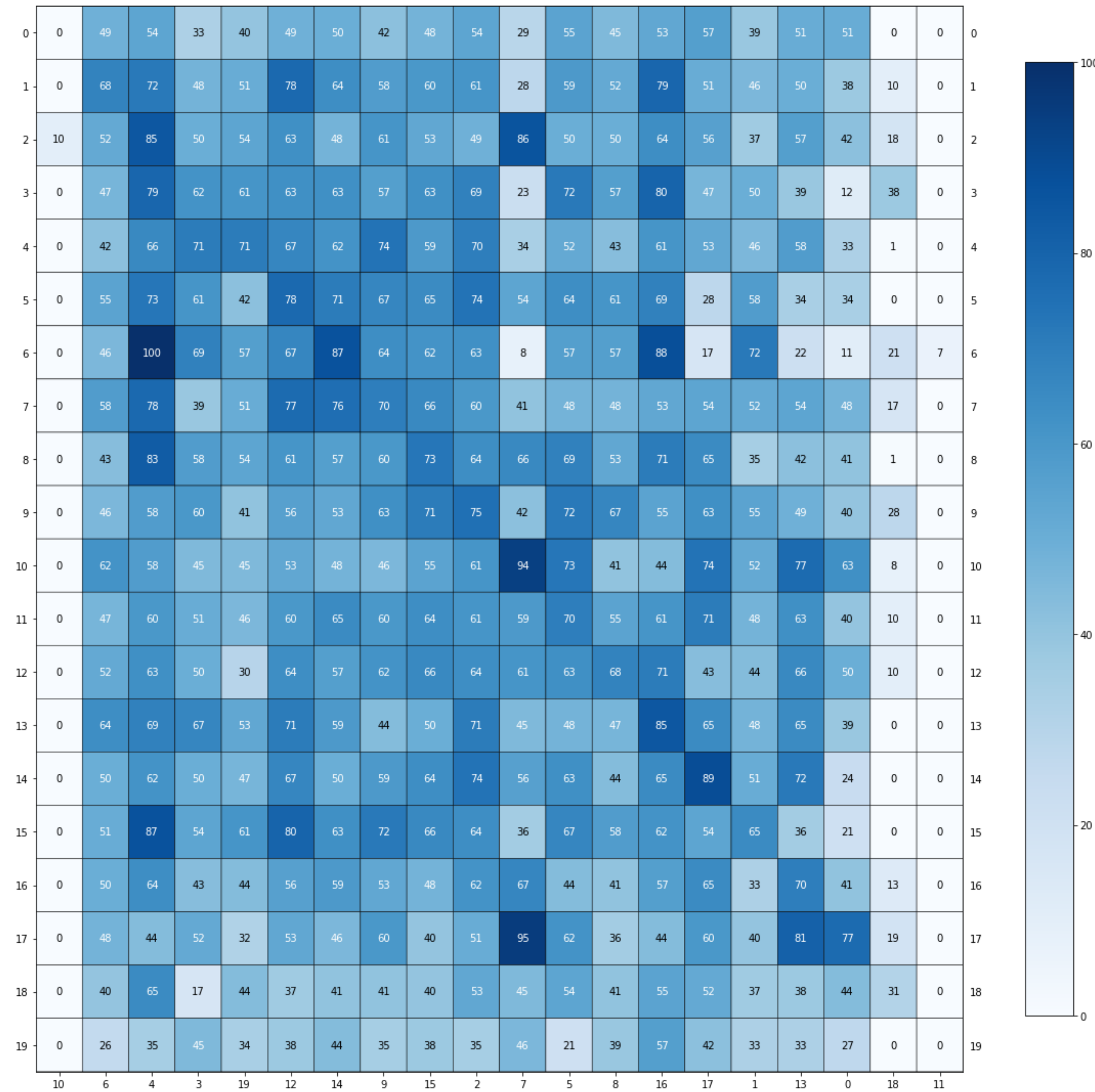


Question 11

- Using Euclidean Distance

- For n_components = [5,20,200], we choose n_components = 20

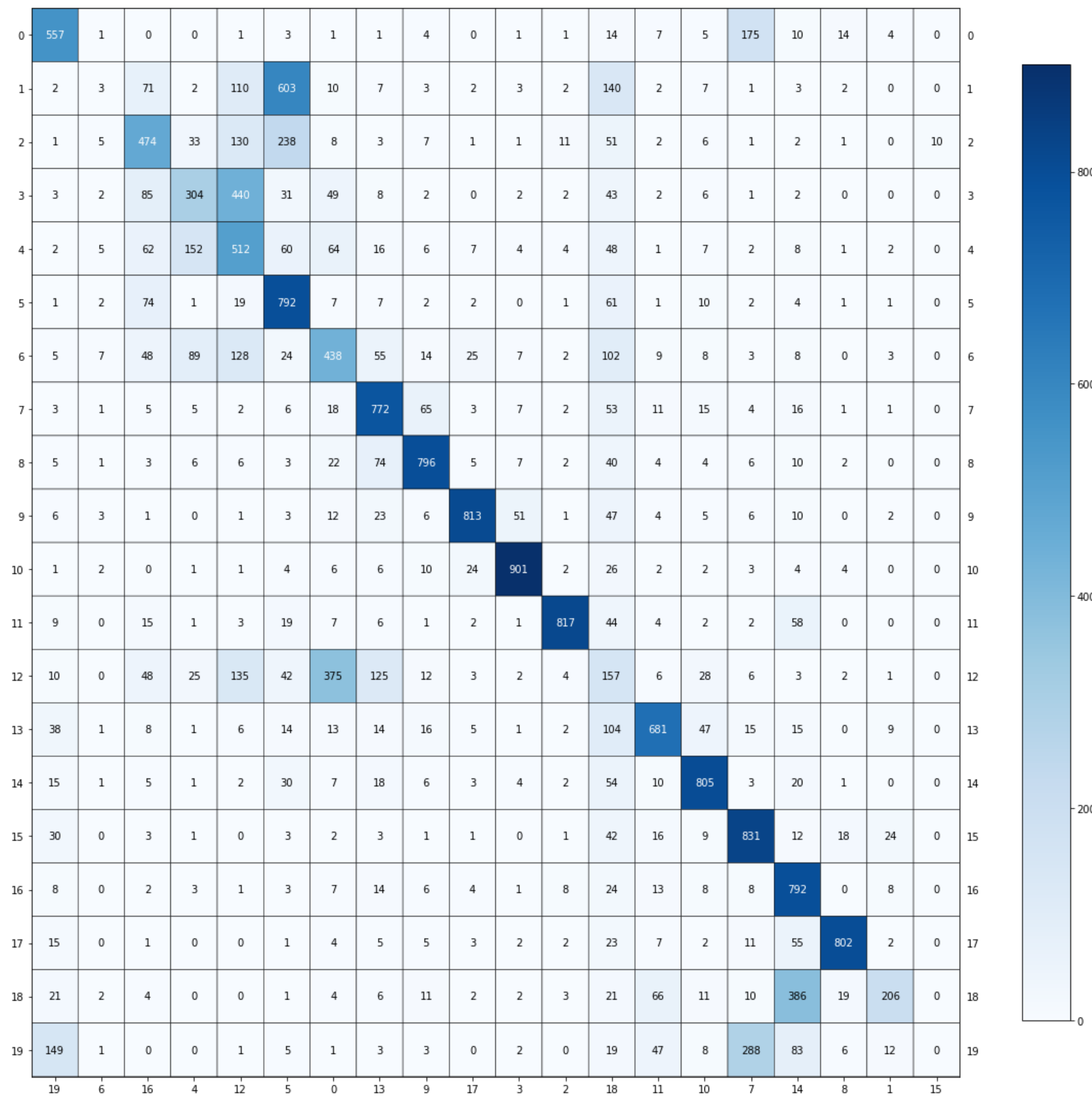
- ==== n_components: 5 ====
- Homogeneity: 0.010000124783356683
- Completeness: 0.010777826875441106
- V-measure: 0.010374421446128561
- Adjusted Rand: 0.0016296507619408358
- Adjusted mutual info: 0.007077890095935255
- ==== n_components: 20 ====
- Homogeneity: 0.010769160820139678
- Completeness: 0.011583480571548786
- V-measure: 0.011161487624308858
- Adjusted Rand: 0.001409146474814078
- Adjusted mutual info: 0.007870434974262288
- ==== n_components: 200 ====
- Homogeneity: 0.007232250179959192
- Completeness: 0.0076771208546437385
- V-measure: 0.007448048419172545
- Adjusted Rand: 0.0014390199321171735
- Adjusted mutual info: 0.004302396679702002



- Using Cosine
    - For n_components = [5,20,200], we choose n_components = 20

    - `==== n_components: 5 ====`
    - `Homogeneity:            0.5611270649073593`
    - `Completeness:           0.5796345993293621`
    - `V-measure:              0.570230700482139`
    - `Adjusted Rand:          0.4391848180501076`
    - `Adjusted mutual info:   0.5688089018119032`
    - `==== n_components: 20 ====`
    - `Homogeneity:            0.5629311494502759`
    - `Completeness:           0.5873873534964182`

- V-measure: 0.574899277424936
- Adjusted Rand: 0.45065869176728857
- Adjusted mutual info: 0.5734844834119995
- ==== n_components: 200 ====
- Homogeneity: 0.5695623708583318
- Completeness: 0.5958570228831612
- V-measure: 0.5824130617157027
- Adjusted Rand: 0.4484053805159569
- Adjusted mutual info: 0.5810229819906624



Question 12

- Based on the contingency matrix from question 11, we can see that using Euclidean distance performs poorly whereas by using cosine, we can see a much more clear diagonal line.

Question 13

- We compare each method with different parameter selection. Detailed results can be found in our jupyter notebook. Here we only show the best parameter results for each method.
- We set **n_components = 20** for KMeans function for all the four following approaches.
- After comparing the adjusted rand index, **UMAP** is the best approach for the K-Means clustering task on the 20-class text data.

- Sparse Representation :
    - `Homogeneity:             0.34790859768457`
    - `Completeness:            0.39677714833899`
    - `V-measure:               0.37073942131701093`
    - **`Adjusted Rand:           0.12210793219608113`**
    - `Adjusted mutual info:    0.36856401463430694`
- NMF (r = 5) :
    - `Homogeneity:             0.26968243804071834`
    - `Completeness:            0.32136572364263605`
    - `V-measure:               0.29326439865012643`
    - **`Adjusted Rand:           0.0884926868130358`**
    - `Adjusted mutual info:    0.2907395776701989`
- SVD (r = 100) :
    - `Homogeneity:             0.35213081962046117`
    - `Completeness:            0.4334267387430322`
    - `V-measure:               0.3885721959749402`
    - **`Adjusted Rand:           0.11948852937487153`**
    - `Adjusted mutual info:    0.3863690421217935`
- UMAP (n_components = 20, metric = 'cosine') :
    - `Homogeneity:             0.5629311494502759`
    - `Completeness:            0.5873873534964182`
    - `V-measure:               0.574899277424936`
    - **`Adjusted Rand:           0.45065869176728857`**
    - `Adjusted mutual info:    0.5734844834119995`

Question 14

- UMAP parameters: n_components = 20, metric = 'cosine'
- Ward:

    - `Homogeneity:             0.559171437894619`

- Completeness: 0.5931374880726187
- V-measure: 0.5756538626069768
- Adjusted Rand: 0.43301259088929217
- Adjusted mutual info: 0.5742292249253264

- Single:

    - Homogeneity: 0.005462116819168942
    - Completeness: 0.23702279614125754
    - V-measure: 0.010678158781292099
    - Adjusted Rand: 9.375957767499856e-06
    - Adjusted mutual info: 0.005714325642047656

- **We can observe that 'ward' performs better than 'single'.**

Question 15

- DBSCAN:
    - We experiment on the hyperparameter "eps".
    - eps = [0.1, 0.3, 0.5, 5] reaches the best result when **eps = 0.3.**
    - ==== DBSCAN eps = 0.1 ====
    - Homogeneity: 0.42381395664827637
    - Completeness: 0.34644828520616694
    - V-measure: 0.3812457901966913
    - Adjusted Rand: 0.007227676224187832
    - Adjusted mutual info: 0.3151205280085486
    - ==== DBSCAN eps = 0.3 ====
    - Homogeneity: 0.46616919918527683
    - Completeness: 0.5338539200765448
    - V-measure: 0.49772100186585105
    - Adjusted Rand: 0.25010682757064484
    - Adjusted mutual info: 0.48326378343030063
    - ==== DBSCAN eps = 0.5 ====
    - Homogeneity: 0.11541745319628637
    - Completeness: 0.5540264372727398
    - V-measure: 0.19103713187562596
    - Adjusted Rand: 0.02041413479283596
    - Adjusted mutual info: 0.18289747176681573
    - ==== DBSCAN eps = 5 ====
    - Homogeneity: 0.0007970663392060461
    - Completeness: 0.16274015024007613
    - V-measure: 0.0015863630127374007
    - Adjusted Rand: -2.107260284165674e-06

    - Adjusted mutual info: 0.0008481663845411296

- HDBSCAN
    - We experiment on the hyperparameter “min_cluster_size”.
    - min_cluster_size = [100, 170, 200] reaches the best result when **min_cluster_size = 200** comparing the adjusted rand index.
    - We use min_cluster_size = 100 in this question.

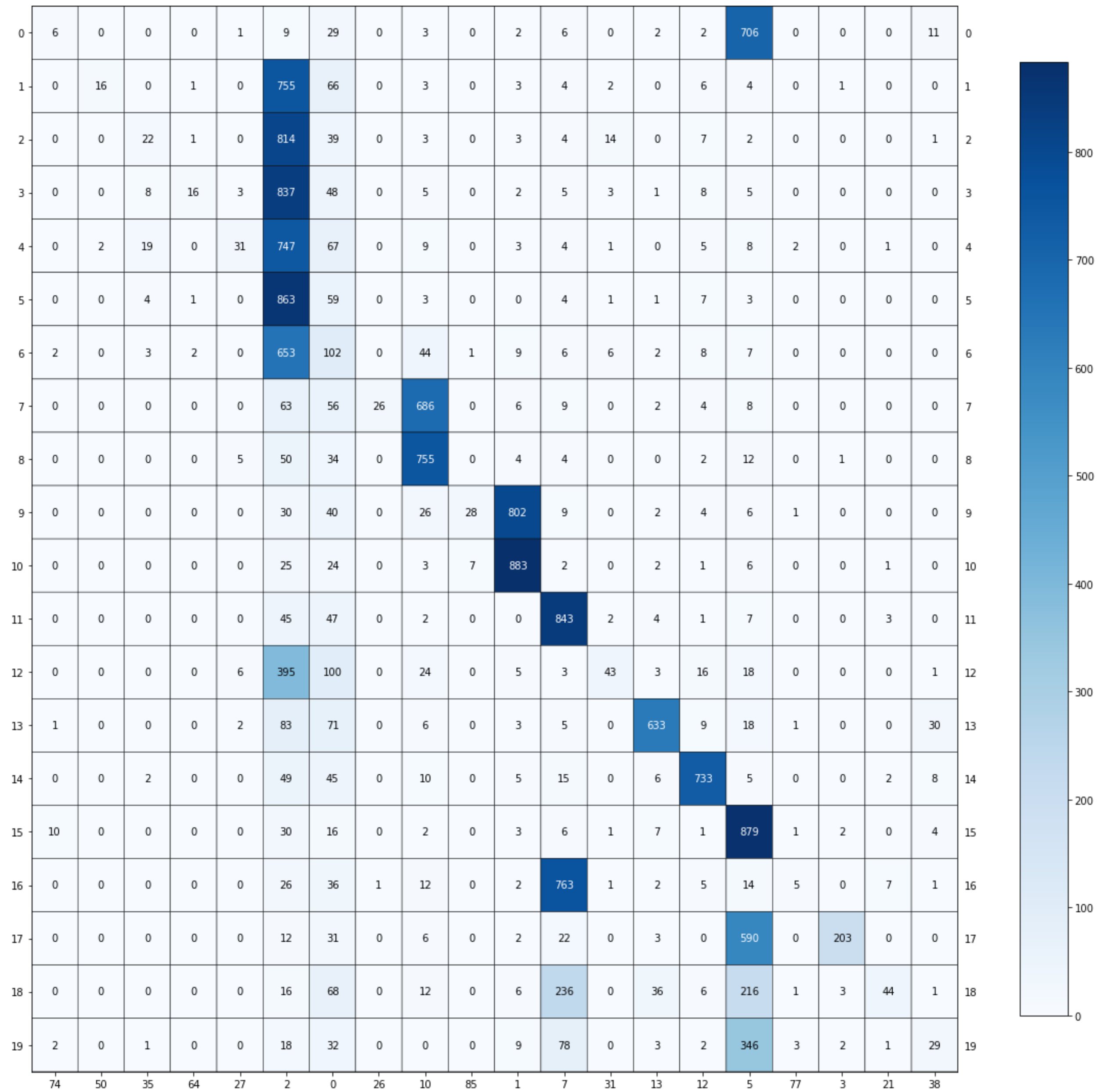
    - ==== HDBSCAN min_cluster_size = 100 ====
    - Homogeneity: 0.42533582661541225
    - Completeness: 0.6185199576492612
    - V-measure: 0.5040518075975402
    - Adjusted Rand: 0.2125326667023847
    - Adjusted mutual info: 0.5029521923361047
    - ==== HDBSCAN min_cluster_size = 170 ====
    - Homogeneity: 0.42437314223474704
    - Completeness: 0.6137889329829664
    - V-measure: 0.5018012974597816
    - Adjusted Rand: 0.21355033403002427
    - Adjusted mutual info: 0.5006991637167756
    - ==== HDBSCAN min_cluster_size = 200 ====
    - Homogeneity: 0.42041044830587443
    - Completeness: 0.6155440699375601
    - V-measure: 0.4995994589284899
    - Adjusted Rand: 0.21879853457021733
    - Adjusted mutual info: 0.49858984875380913

Question 16

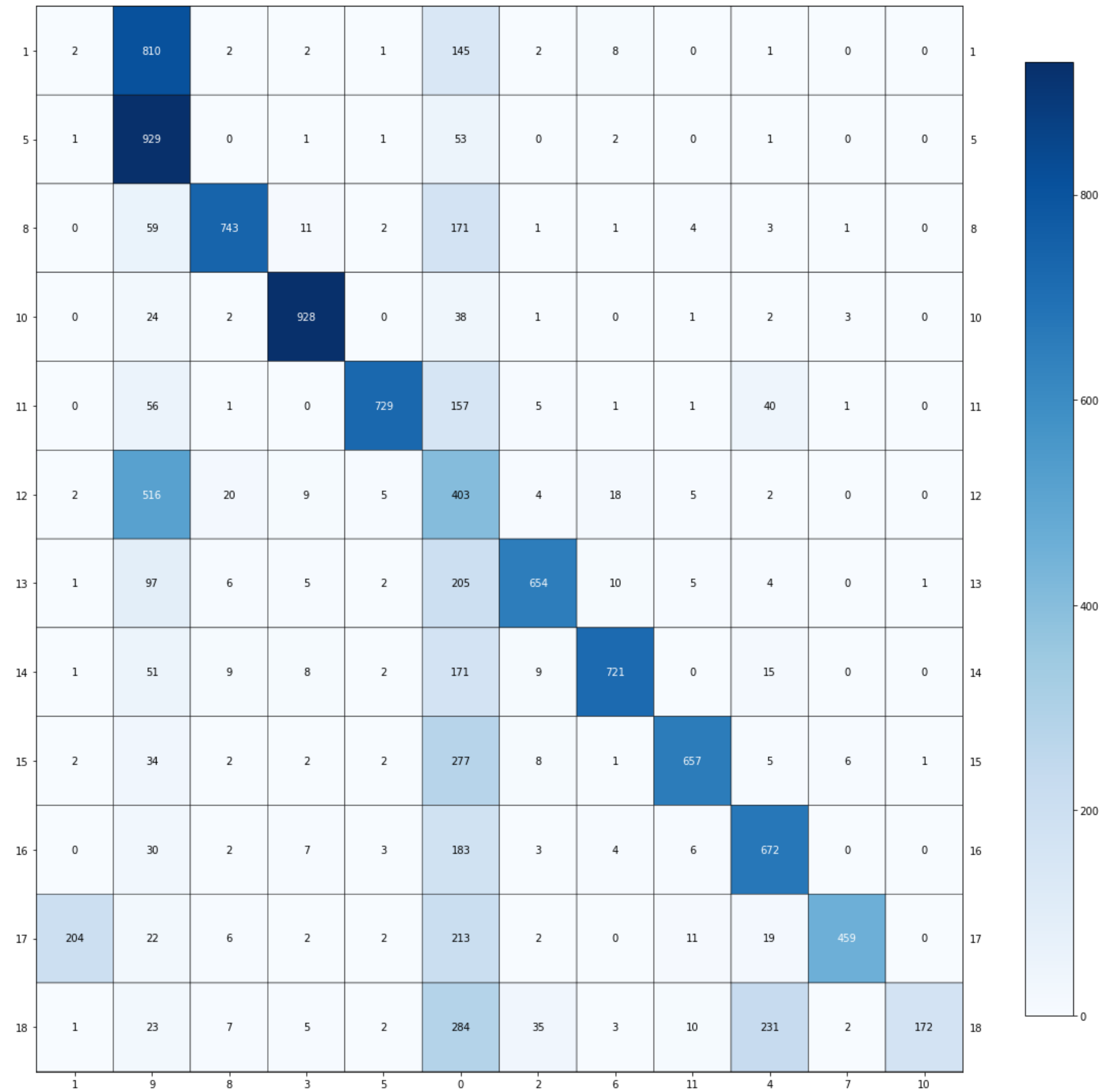
- By comparing all **adjusted index scores** in Q15 using min_cluster_size = 100 for HDBSCAN, **DBSCAN with eps = 0.3 reached the best result.**

- For the DBSCAN model(eps = 0.3), there are 155 clusters.

- **Noisy samples** are given the label -1.



- For reference we also plot HDBSCAN contingency matrix with min_cluster_size = 100.
- There are 11 clusters.



Question 17

- We’ve made a table according to our results. Details can be found in the code.
- According to the adjusted rand index matrix, it seems that **UMAP** as dimensionality reduction method with **K-means clustering** method work best together for 20-class text data.

Module	Alternatives	Hyperparameters	Adjusted Rand Index
Dimensionality Reduction	None (with K-Means)	N/A	0.1221
	SVD (with K-Means)	r = [5, 20, 100, 200]	r = 5 : 0.1284 r = 20 : 0.1046 r = 100 : 0.1195 r = 200 : 0.0928
	NMF (with K-Means)	r = [5, 20, 100, 200]	r = 5 : 0.0885 r = 20 : 0.0785 r = 100 : 0.0094 r = 200 : 0.0065
	UMAP (with K-Means)	n_components = [5, 20, 200] metric = ‘cosine’	n_components = 5 : 0.4392 **n_components = 20 : 0.4507** n_components= 200 : 0.4484
Clustering	**K-Means (with UMAP)**	k = [10, 20, 50]	k = 10 : 0.3349 **k = 20 : 0.4592** k = 50 : 0.3929
	Agglomerative Clustering (with UMAP)	n_clusters = [20] linkage criteria = ‘ward’	n_clusters = 20 : 0.4330
	DBSCAN (with UMAP)	eps = [0.1, 0.3, 0.5, 5]	eps = 0.1 : 0.0072 eps = 0.3 : 0.2501 eps = 0.5 : 0.0204 eps = 0.5 : -2.107e-06
	HDBSCAN (with UMAP)	min_cluster_size = [100, 170, 200]	min_cluster_size = 100 : 0.2125 min_cluster_size = 170 : 0.2136 min_cluster_size = 200 : 0.2188

Question 18 (Bonus)

Question 19

- Networks can generally learn global representation of an image, and such a representation has discriminative power for a custom dataset. For example, the network can learn color representation regardless of the actual label. Lions and tigers have different labels during training, the network will first learn the low-level color information of the images, then learn to use more dense features like shape to discriminate between the two species. Although the high-level feature learned by the network will no longer be valid in customer dataset, the low-level feature will also help to have discriminative power.

Question 20

- The helper code first uses VGG-16 feature layers to extract raw features, then uses a pooling layer to compress the feature, and finally performs a linear transformation to obtain the final feature representation.

Question 21

- The image size is 224*224, in total **50176** pixels. The feature dimension is **4096** for one image.

Question 22

- Compared with sparse TF-IDF features in text, features extracted from image are **dense**, since the feature vectors are 100% non-zero or non-empty in our experiment.

Question 23

- From the t-NSE plot we can observe that all five types of flowers are clearly classified and clustered. There are intersections between clusters which indicate some of the images are incorrectly classified, however the intersection area is very small and most instances are clustered very well.

Question 24

- UMAP + K-Means is the best combination together.

Module	Alternatives	Hyperparameters	Adjusted Rand Index
Dimensionality Reduction	None (with K-Means)	N/A	0.1898
	SVD (with K-Means)	r = 50	0.1949
	UMAP (with K-Means)	n_components = 50	0.4660
	Autoencoder (with K-Means)	n_features = 50	0.1660
Clustering	**K-Means (with UMAP)**	k = 50	0.4651
	Agglomerative Clustering (with UMAP)	n_clusters = 5	0.4591
	HDBSCAN (with UMAP)	min_cluster_size =170 Min_samples = None	0.0941

Question 25

- We use UMAP as our dimension reduction method. By using reduced-dimension features, the accuracy **drops from 91.0% to 85.8%**. Our model suffers from reduced-dimension features but **we do not think it is significant**. We also try different reduced-dimensions with UMAP and we find the accuracy fluctuates along the dimension size, and achieves highest accuracy at **dimension=10**.
- Correlating with Question 24, we can see that **VGG features are rich enough in information about the data classes** -- both the original VGG features and the reduced-dimension features achieve high accuracy scores for the MLP classifier. Yet on the other hand, clustering results obtained for the same features in Question 24 rely highly on dimension reduction.