Project 2

Marc Hernandez, Tilboon Elberier

In [70]:
```
# import functions needed for problems 1-18
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.decomposition import TruncatedSVD
from sklearn.decomposition import NMF
from sklearn.cluster import KMeans
from sklearn.metrics.cluster import contingency_matrix, homogeneity_score, v_measur
from sklearn.cluster import AgglomerativeClustering

from scipy import stats
from scipy.optimize import linear_sum_assignment
import importlib
import plotmat
import hdbscan

import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
# Import newsgroup data for TFIDF Transformer
cata = ['comp.graphics', 'comp.os.ms-windows.misc', 'comp.sys.ibm.pc.hardware', 'co
        'rec.autos', 'rec.motorcycles', 'rec.sport.baseball', 'rec.sport.hockey']
newsgroups_train = fetch_20newsgroups(subset='all', categories=cata, shuffle=True,
```

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

In [3]:
```
# Find the Shape of the
countVectorizer = CountVectorizer(min_df=3, stop_words='english')
tfidf_transformer = TfidfTransformer()
newsgroups_counts = countVectorizer.fit_transform(newsgroups_train.data)
newsgroups_tfidf = tfidf_transformer.fit_transform(newsgroups_counts)
print("ANSWER QUESTION 1: The shape of the TF-IDF matrix is ", newsgroups_tfidf.sha
```

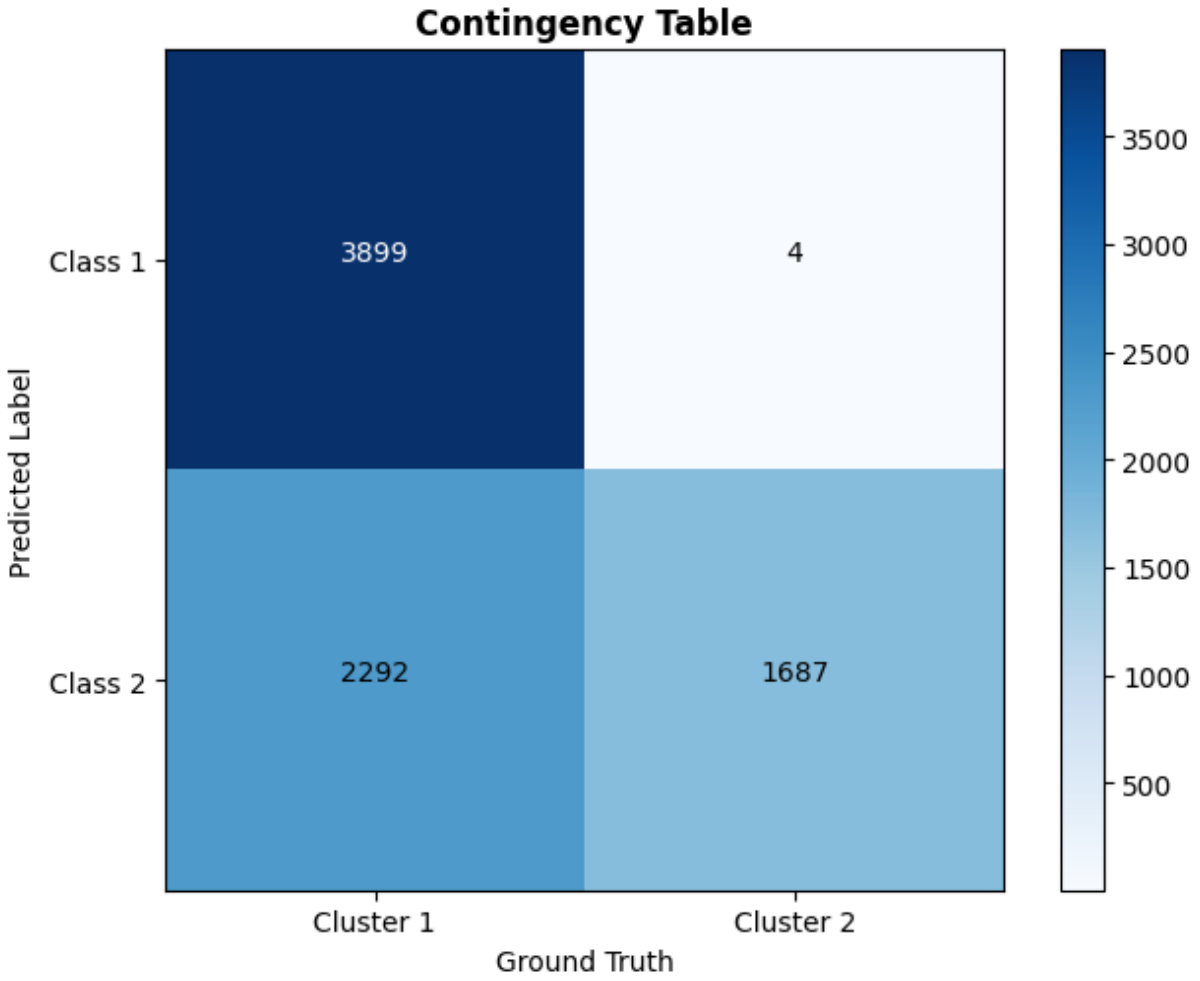
```
ANSWER QUESTION 1: The shape of the TF-IDF matrix is  (7882, 27768)
```

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?

In [4]:
```
km = KMeans(n_clusters=2, random_state=0, max_iter=1000, n_init=30)
y_pred = km.fit_predict(newsgroups_tfidf)
y_true = [int(i/4) for i in newsgroups_train.target]
```

```
cluster_class_names=['Class 1', 'Class 2']
actual_class_names=['Cluster 1', 'Cluster 2']

cm = contingency_matrix(y_true, y_pred)
print("Contingency table: \n", cm)
plt.imshow(cm, cmap=plt.cm.Blues)
plt.colorbar()
plt.xticks(range(len(actual_class_names)), actual_class_names)
plt.yticks(range(len(actual_class_names)), cluster_class_names)


for i in range(cm.shape[0]):
    for j in range(cm.shape[1]):
        if np.argmax(cm) == i*cm.shape[1]+j:
            plt.text(j, i, cm[i, j], horizontalalignment="center", color='w')
        else:
            plt.text(j, i, cm[i, j], horizontalalignment="center", color='k')

plt.tight_layout()
plt.xlabel('Ground Truth')
plt.ylabel('Predicted Label')
plt.title('Contingency Table', fontweight='bold')
plt.show()
```

```
Contingency table:
 [[3899    4]
 [2292 1687]]
```



No, the contingency matrix doesn't have to be square shaped because the matrix itself represents the frequency of occurances between our predictions at the group truth label. The shape of the contringency matrix could be rectangular.

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

In [5]:
```
print("Below are the 5 clustering Evaluation Metrics described within the project 2
def get_metrics(y_true, y_pred):
    diction = {}
    metrics = [homogeneity_score, completeness_score, v_measure_score, adjusted_ran

    for i in metrics:
        diction[i.__name__] = i(y_true, y_pred)

    df = pd.DataFrame(diction, index=[0]).T
    df.reset_index(inplace=True)
    df.rename(columns={'index':'Evaluation', 0: 'Result'}, inplace=True)

    return df

get_metrics(y_true, y_pred)
```

Below are the 5 clustering Evaluation Metrics described within the project 2 documentation

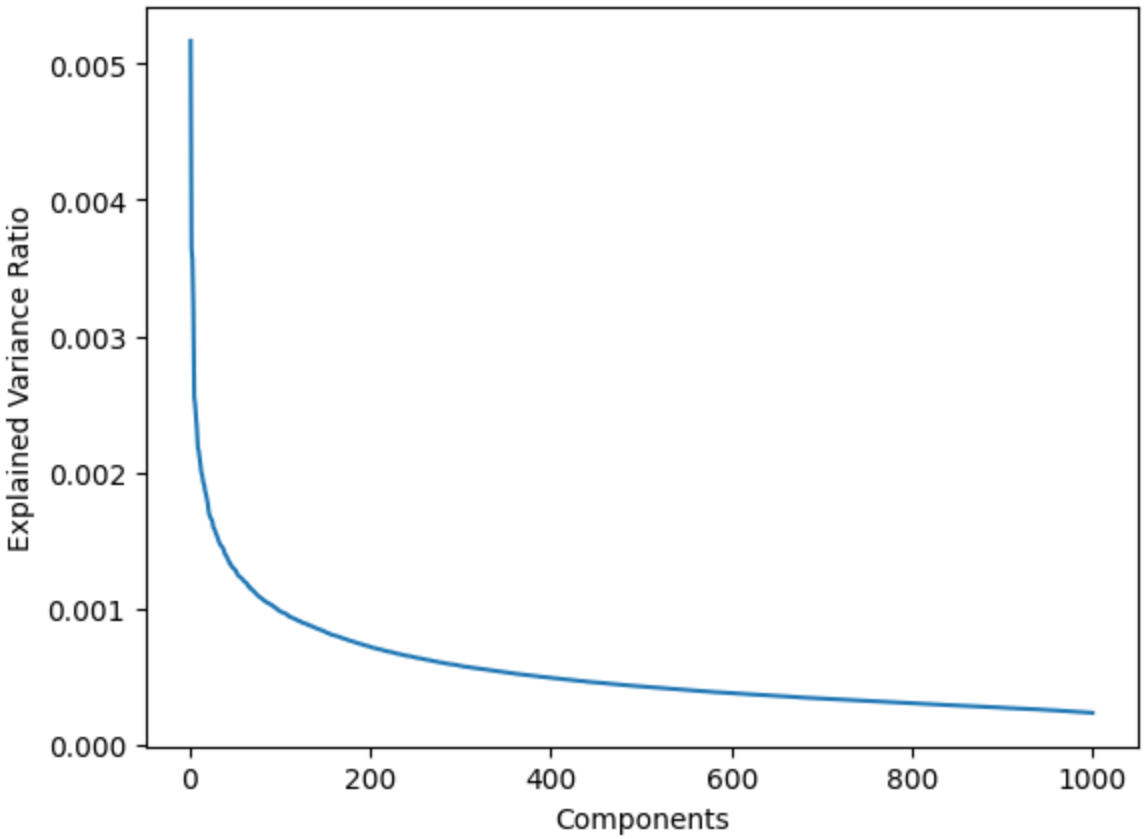
Out[5]:

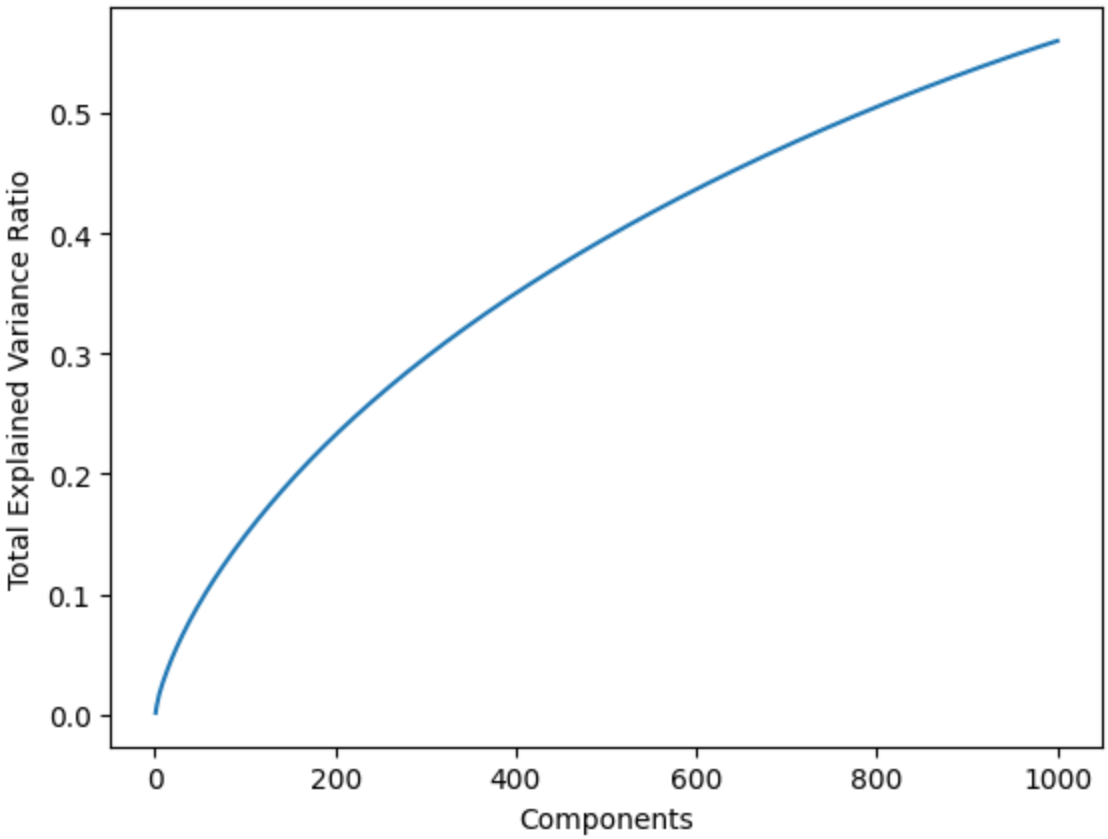
	Evaluation	Result
0	homogeneity_score	0.247946
1	completeness_score	0.330542
2	v_measure_score	0.283348
3	adjusted_rand_score	0.174133
4	adjusted_mutual_info_score	0.283273

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.

In [6]:
```
svd = TruncatedSVD(n_components=1000, random_state=42)
X_svd = svd.fit_transform(newsgroups_tfidf)

plt.figure()
plt.plot(np.arange(1000) + 1, sorted(svd.explained_variance_ratio_, reverse=True))
plt.xlabel("Components")
plt.ylabel("Explained Variance Ratio")

plt.figure()
plt.plot(np.arange(1000) + 1, np.cumsum(svd.explained_variance_ratio_))
plt.xlabel("Components")
plt.ylabel("Total Explained Variance Ratio")
```

Out[6]:
```
Text(0, 0.5, 'Total Explained Variance Ratio')
```





QUESTION 5: Let r be the dimension that we want to reduce the data to (i.e. n components). Try r = 1 − 10, 20, 50, 100, 300, and plot the 5 measure scores v.s. r for both SVD and NMF.

Report a good choice of r for SVD and NMF respectively.

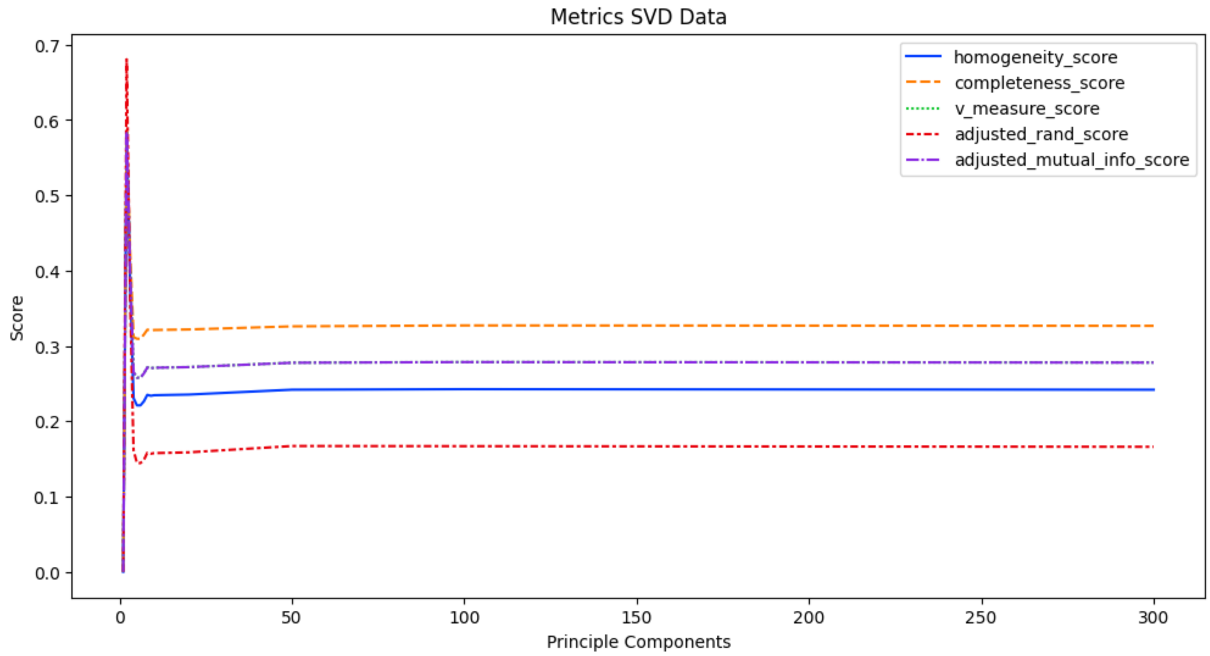
Note: In the choice of r, there is a trade-off between the information preservation, and better performance of k-means in lower dimensions.

In [7]:
```
components = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 20, 50, 100, 300]
km = KMeans(n_clusters=2, random_state=0, max_iter=1000, n_init=30)

result = []
for r in components:
    svd = TruncatedSVD(n_components=r, random_state=42).fit_transform(newsgroups_tf
    preds = km.fit_predict(svd)
    result.append(get_metrics(y_true, preds)['Result'].tolist())

metrics = get_metrics(y_true, preds)['Evaluation'].tolist()
metrics_df = pd.DataFrame(result, columns=metrics, index=components)
metrics_df
```

Out[7]:

	homogeneity_score	completeness_score	v_measure_score	adjusted_rand_score	adjuste
1	0.000284	0.000289	0.000286	0.000317	
2	0.583085	0.585240	0.584160	0.680448	
3	0.382045	0.425804	0.402739	0.364320	
4	0.230775	0.311849	0.265255	0.161654	
5	0.221171	0.309568	0.258008	0.144576	
6	0.221171	0.309568	0.258008	0.144576	
7	0.226504	0.315106	0.263557	0.148658	
8	0.235158	0.321620	0.271676	0.158405	
9	0.233915	0.320684	0.270512	0.156993	
10	0.234625	0.321219	0.271177	0.157799	
20	0.235514	0.321888	0.272009	0.158809	
50	0.241986	0.326035	0.277792	0.167212	
100	0.242668	0.327276	0.278692	0.167004	
300	0.241949	0.326734	0.278022	0.166176	

In [8]:
```
plt.figure(figsize=(12, 6))
sns.set_palette("bright")
g = sns.lineplot(data=metrics_df)
g.set_xlabel('Principle Components')
g.set_ylabel('Score')
g.set_title('Metrics SVD Data')
plt.show()
```



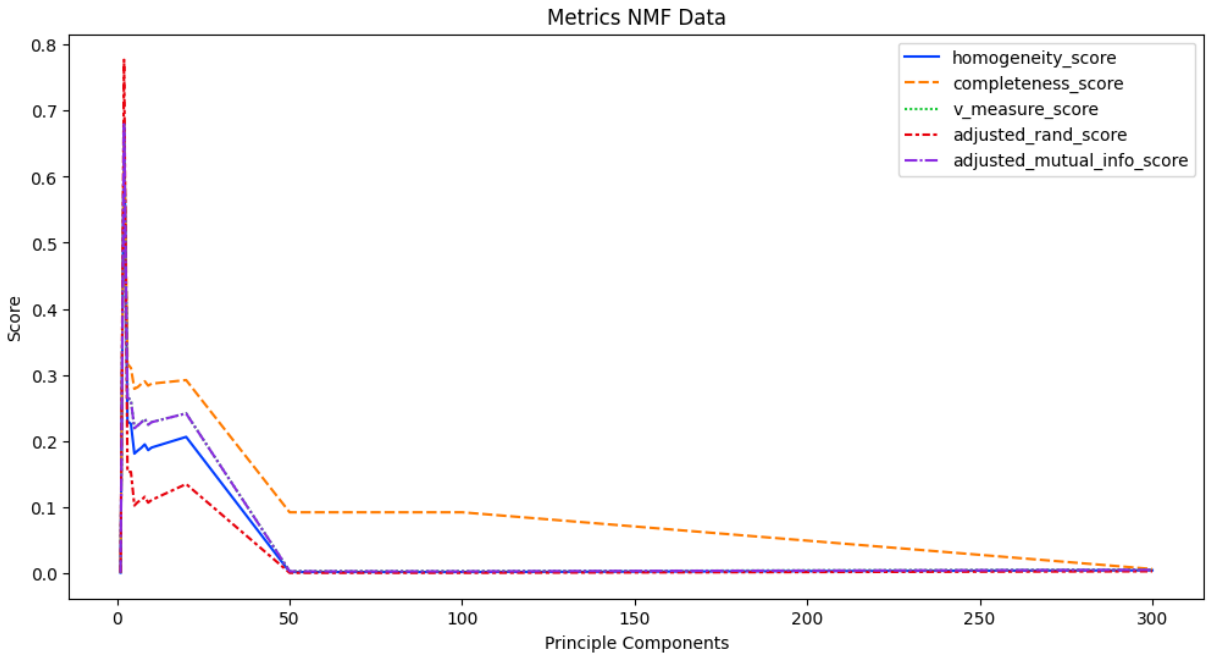
In [9]:
```
result = []
for r in components:
    nmf = NMF(n_components=r, random_state=42).fit_transform(newsgroups_tfidf)
    preds = km.fit_predict(nmf)
    result.append(get_metrics(y_true, preds)['Result'].tolist())

metrics = get_metrics(y_true, preds)['Evaluation'].tolist()
metrics_df = pd.DataFrame(result, columns=metrics, index=components)
metrics_df
```

```
c:\Users\astro\repositories\UCLA\ECE 219\repo\.venv\Lib\site-packages\sklearn\decomp
osition\_nmf.py:1742: ConvergenceWarning: Maximum number of iterations 200 reached.
Increase it to improve convergence.
  warnings.warn(
c:\Users\astro\repositories\UCLA\ECE 219\repo\.venv\Lib\site-packages\sklearn\decomp
osition\_nmf.py:1742: ConvergenceWarning: Maximum number of iterations 200 reached.
Increase it to improve convergence.
  warnings.warn(
c:\Users\astro\repositories\UCLA\ECE 219\repo\.venv\Lib\site-packages\sklearn\decomp
osition\_nmf.py:1742: ConvergenceWarning: Maximum number of iterations 200 reached.
Increase it to improve convergence.
  warnings.warn(
```

Out[9]:

	homogeneity_score	completeness_score	v_measure_score	adjusted_rand_score	adjuste
1	0.000284	0.000289	0.000286	0.000317	
2	0.679048	0.680132	0.679590	0.777018	
3	0.229343	0.316484	0.265957	0.152797	
4	0.225779	0.310382	0.261406	0.152996	
5	0.180631	0.278709	0.219199	0.101956	
6	0.185311	0.281480	0.223489	0.107376	
7	0.189325	0.286194	0.227893	0.109886	
8	0.194655	0.290243	0.233028	0.115338	
9	0.185850	0.283550	0.224533	0.106380	
10	0.189823	0.286573	0.228373	0.110392	
20	0.205946	0.291986	0.241533	0.134526	
50	0.001288	0.091729	0.002540	0.000055	
100	0.001288	0.091729	0.002540	0.000055	
300	0.003918	0.005839	0.004689	0.002361	

In [10]:
```
plt.figure(figsize=(12, 6))
sns.set_palette("bright")
g = sns.lineplot(data=metrics_df)
g.set_xlabel('Principle Components')
g.set_ylabel('Score')
g.set_title('Metrics NMF Data')
plt.show()
```



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

From our graphs we can observe that the influence of the number of components is non-monotonic, as the resulting score does not increase consistently as the number of components also increases. This is likely due to the curse of dimentionality as, K-means relies on the Euclidean Distance to calculate distances between points and nearby clusters, and as we continue to increase the number of cluster groups (i.e. as the number of components increases) then it will be more difficult for our model to properly capture the information provided by the data.

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

Yes, on average the measurements observed using SVD average do preform better, however, NMF is close and could be argued that it doesn't preform better than the baseline implementation in question 3

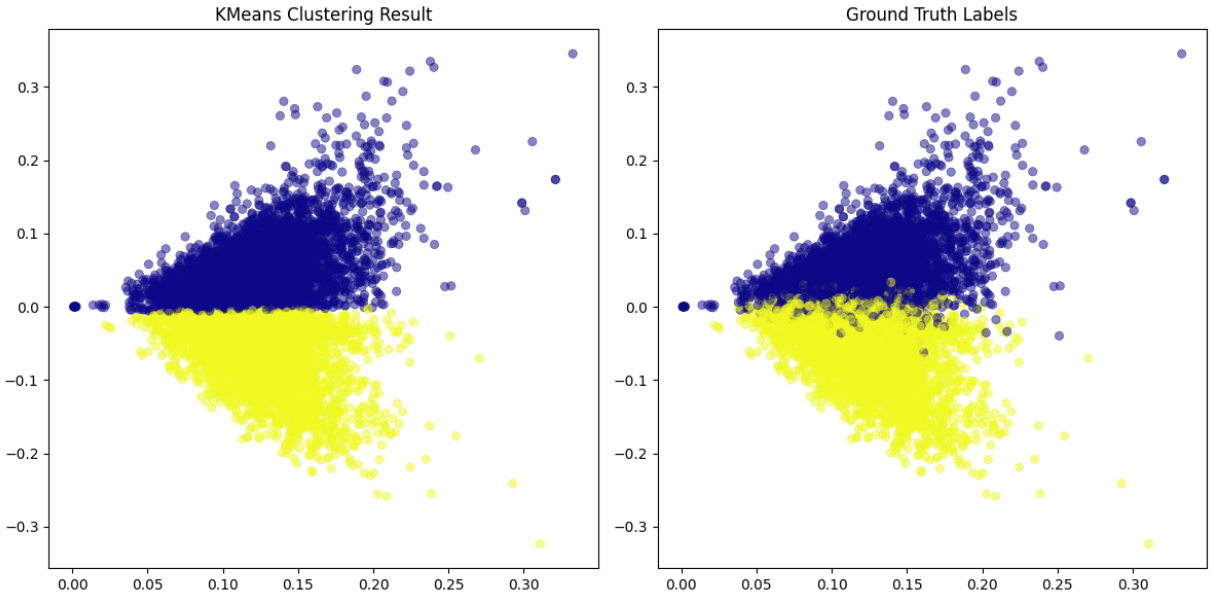
QUESTION 8: Visualize the clustering results for:

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

To recap, you can accomplish this by first creating the dense representations and then once again projecting these representations into a 2-D plane for visualization.

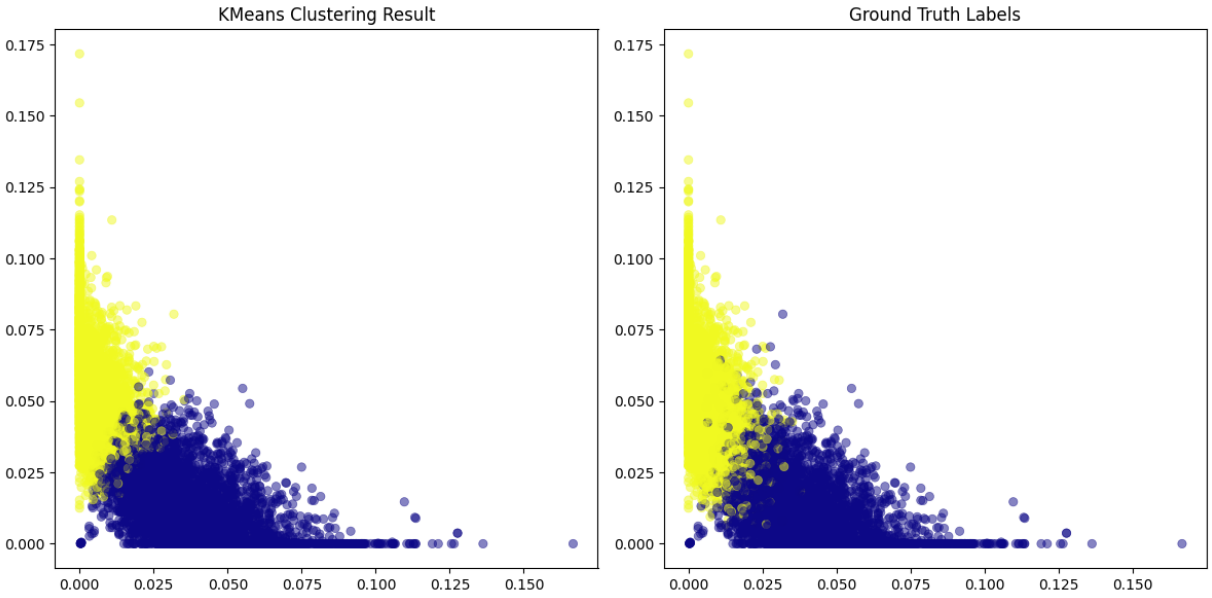
In [11]:
```
# Best number of components was r = 2
svd = TruncatedSVD(n_components=2, random_state=42).fit_transform(newsgroups_tfidf)
svd_pred = km.fit_predict(svd)

plt.gcf().clear()
plt.figure(figsize=(12,6))
plt.subplot(1,2,1)
plt.scatter(svd[:,0], svd[:,1], c=svd_pred, linewidth=0.5, cmap="plasma", alpha=0.5
plt.title('KMeans Clustering Result')
plt.subplot(1,2,2)
plt.scatter(svd[:,0], svd[:,1], c=y_true, linewidth=0.5, cmap="plasma", alpha=0.5)
plt.title('Ground Truth Labels')
plt.tight_layout()
plt.show()
```

```
<Figure size 640x480 with 0 Axes>
```



In [12]:
```
nmf = NMF(n_components=2, random_state=42).fit_transform(newsgroups_tfidf)
nmf_pred = km.fit_predict(nmf)

plt.gcf().clear()
plt.figure(figsize=(12,6))
plt.subplot(1,2,1)
plt.scatter(nmf[:,0], nmf[:,1], c=nmf_pred, cmap="plasma",linewidth=0.5, alpha=0.5)
plt.title('KMeans Clustering Result')
plt.subplot(1,2,2)
plt.scatter(nmf[:,0], nmf[:,1], c=y_true, cmap="plasma", linewidth=0.5, alpha=0.5)
plt.title('Ground Truth Labels')
plt.tight_layout()
plt.show()
```

```
<Figure size 640x480 with 0 Axes>
```



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?

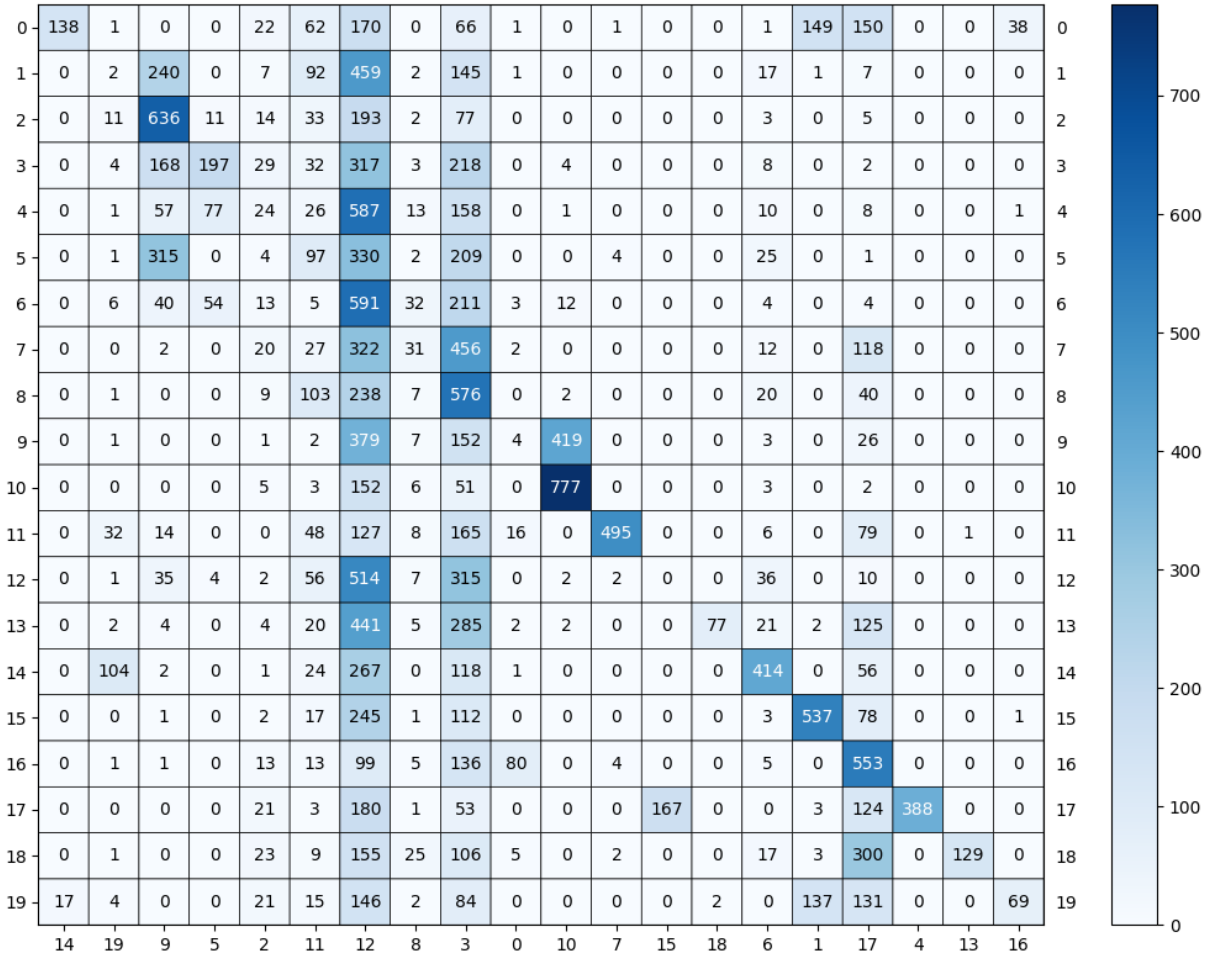
Based on the plots depicted above, one can say that K-means will likely not be the most optimal model for this dataset. Primarily because of the non-isotropic nature of the formed clusters as well as their frequent overlap in both the SVD and NMF graphs. Because of these two issues, this is not an idea distribution for the use of K-Means clustering.

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

There is a mismatch between cluster labels and class labels. For example, the cluster #3 may correspond to the class #8. As a result, the high-value entries of the 20 × 20 contingency matrix can be scattered around, making it messy to inspect, even if the clustering result is not bad.

In [98]:
```
data_full = fetch_20newsgroups(subset='all', shuffle=True, random_state=42)
```

In [99]:
```
count_full = CountVectorizer(min_df=3, stop_words='english')
tfidf_full = TfidfTransformer()
km = KMeans(n_clusters=20, random_state=0, max_iter=1000, n_init=30)
full_counts = count_full.fit_transform(data_full.data)
tfidf_preds = tfidf_full.fit_transform(full_counts)

svd = TruncatedSVD(n_components=20, random_state=42).fit_transform(tfidf_preds)
svd_pred = km.fit_predict(svd)
cm = contingency_matrix(data_full.target, svd_pred)
rows, cols = linear_sum_assignment(cm, maximize=True)
reordered_mat = cm[np.ix_(rows, cols)]
plotmat.plot_mat(reordered_mat, xticklabels=cols, yticklabels=rows, size=(10,8))
```

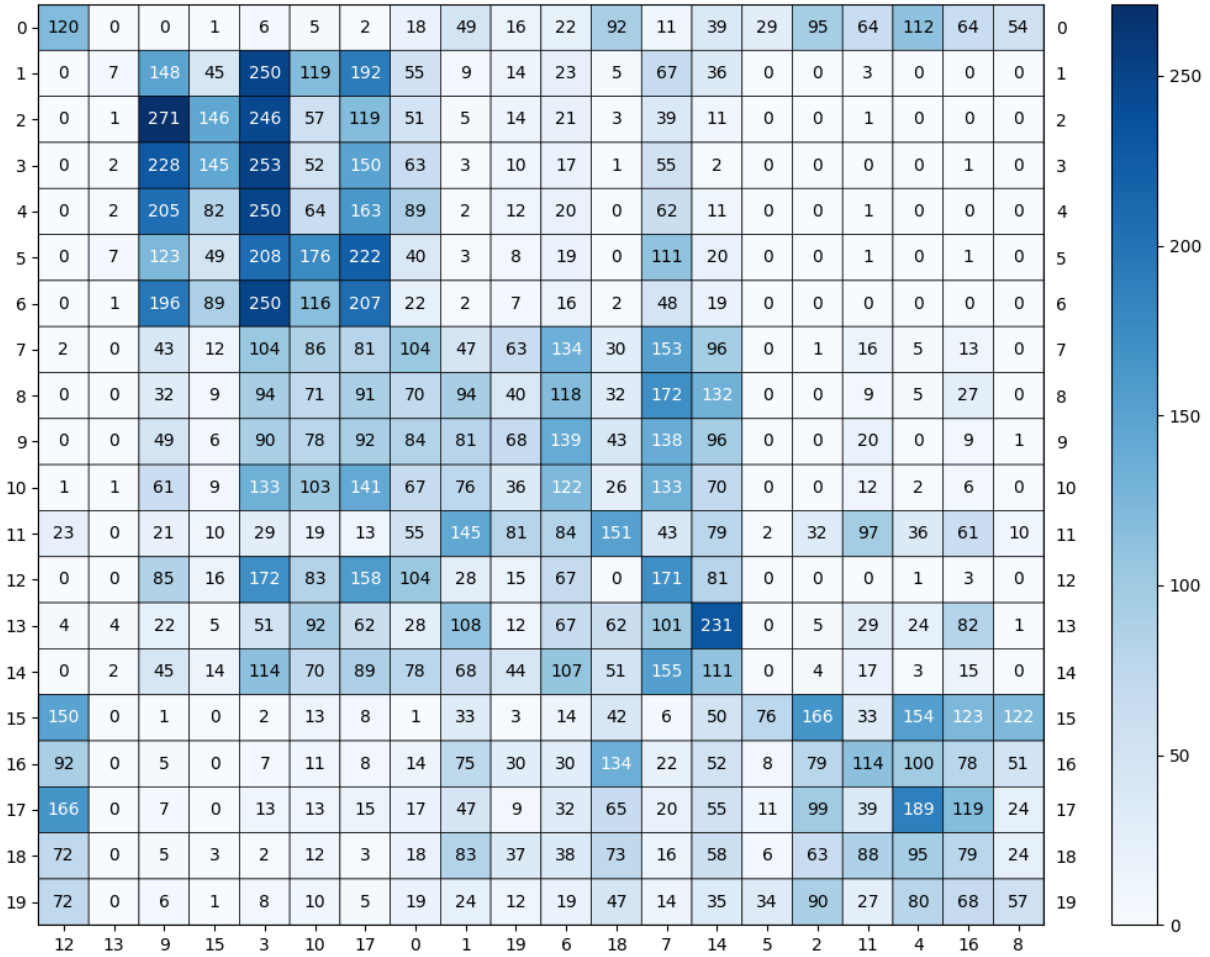


In [47]:
```
get_metrics(data_full.target, y_pred)
```

Out[47]:

	Evaluation	Result
0	homogeneity_score	0.287995
1	completeness_score	0.383499
2	v_measure_score	0.328955
3	adjusted_rand_score	0.093060
4	adjusted_mutual_info_score	0.326445

In [100…
```
count_full = CountVectorizer(min_df=3, stop_words='english')
tfidf_full = TfidfTransformer()
km = KMeans(n_clusters=20, random_state=0, max_iter=1000, n_init=30)
full_counts = count_full.fit_transform(data_full.data)
tfidf_preds = tfidf_full.fit_transform(full_counts)

nmf = NMF(n_components=2, random_state=42).fit_transform(tfidf_preds)
nmf_pred = km.fit_predict(nmf)
cm = contingency_matrix(data_full.target, nmf_pred)
rows, cols = linear_sum_assignment(cm, maximize=True)
reordered_mat = cm[np.ix_(rows, cols)]
plotmat.plot_mat(reordered_mat, xticklabels=cols, yticklabels=rows, size=(10,8))
```



In [46]:
```
get_metrics(data_full.target, nmf_pred)
```

Out[46]:

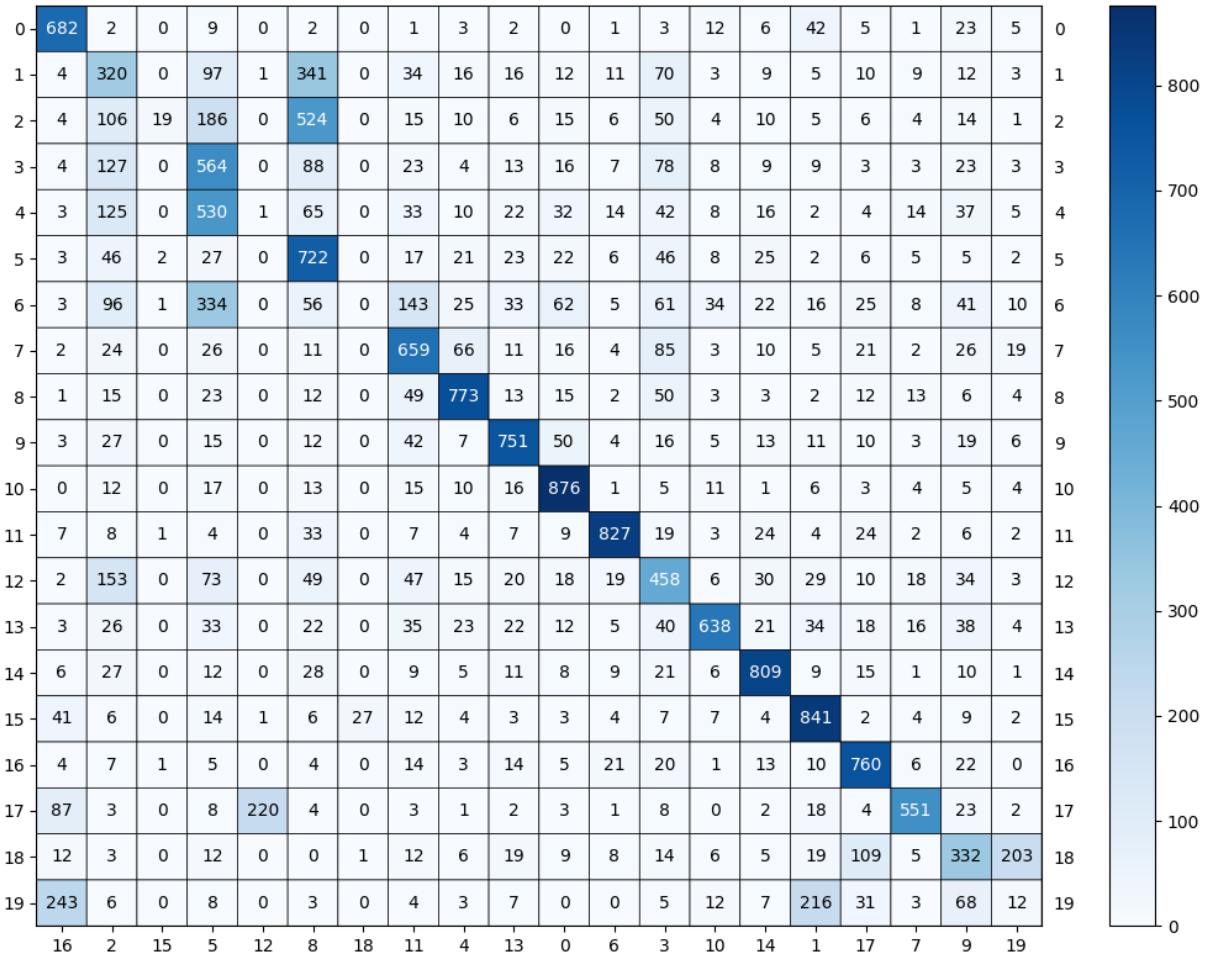
	Evaluation	Result
0	homogeneity_score	0.169491
1	completeness_score	0.179924
2	v_measure_score	0.174552
3	adjusted_rand_score	0.051011
4	adjusted_mutual_info_score	0.171780

QUESTION 11: Reduce the dimension of your dataset with UMAP. Consider the following settings: n components = [5, 20, 200], metric = "cosine" vs. "euclidean". If "cosine" metric fails, please look at the FAQ at the end of this spec.

Report the permuted contingency matrix and the five clustering evaluation metrics for the different combinations (6 combinations).

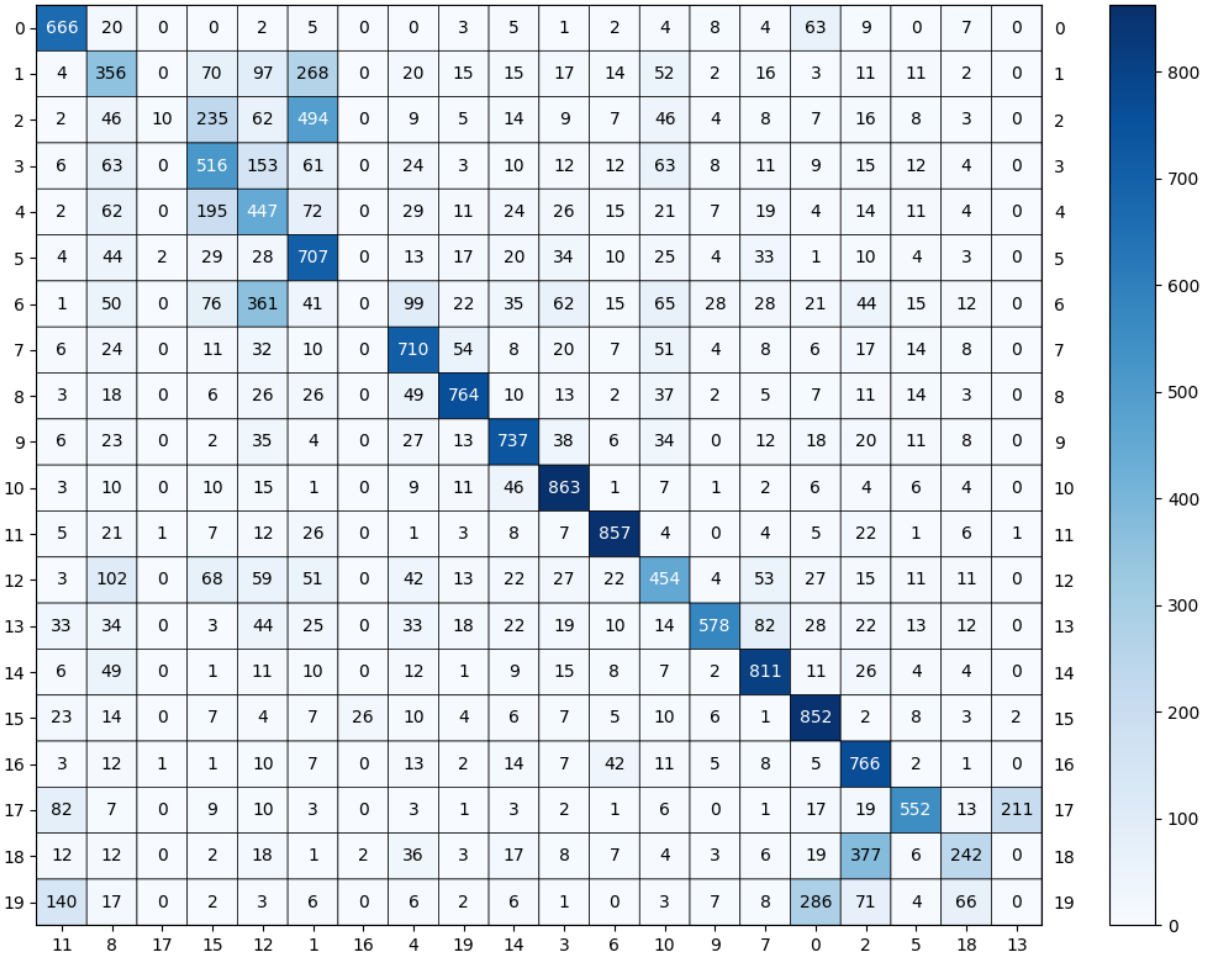
In [63]:
```
import umap.umap_ as umap

components = [5,20,200]
metric = ["cosine", "euclidean"]
```

```
for n in components:
    for i in metric:
        umap_full = umap.UMAP(n_components=n, metric=i).fit_transform(tfidf_preds)
        km = KMeans(n_clusters=20, random_state=0, max_iter=1000, n_init=30)
        y_pred = km.fit_predict(umap_full)

        cm = contingency_matrix(data_full.target, y_pred)
        rows, cols = linear_sum_assignment(cm, maximize=True)
        reordered_mat = cm[np.ix_(rows, cols)]
        plotmat.plot_mat(reordered_mat, xticklabels=cols, yticklabels=rows, size=(1

        print("Number of Components: ", n, "Evaluating Metric: ", i)
        print("Homogeneity : %0.3f" % homogeneity_score(data_full.target, y_pred))
        print("Completeness : %0.3f" % completeness_score(data_full.target, y_pred)
        print("V-measure : %0.3f" % v_measure_score(data_full.target, y_pred))
        print("Adjusted Rand-Index : %.3f"% adjusted_rand_score(data_full.target, y
        print("Adjusted Mutual Information Score : %.3f"% adjusted_mutual_info_scor
```

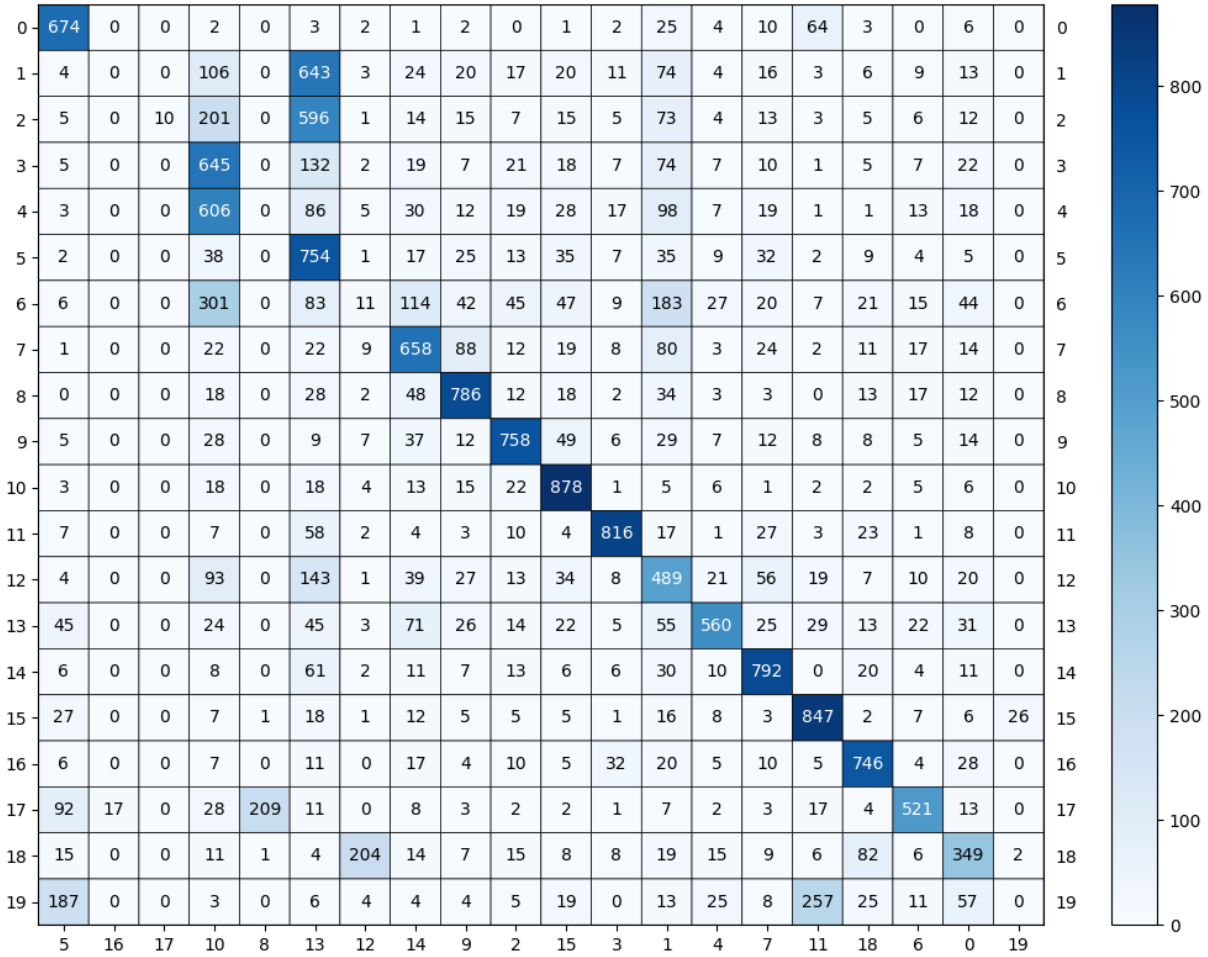



```
Number of Components:  5 Evaluating Metric:  cosine
Homogeneity : 0.494
Completeness : 0.526
V-measure : 0.510
Adjusted Rand-Index : 0.389
Adjusted Mutual Information Score : 0.508
```

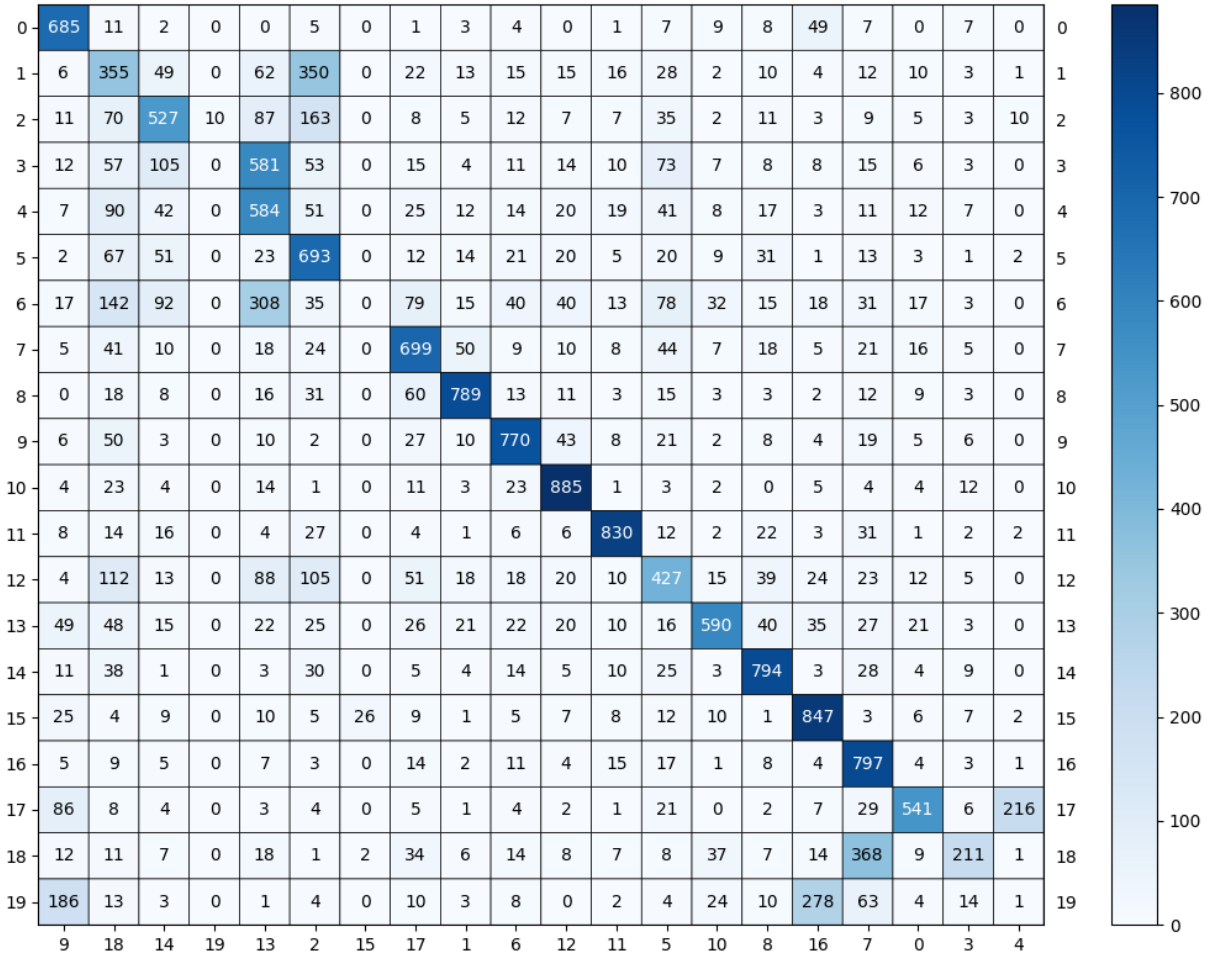
```
c:\Users\astro\repositories\UCLA\ECE 219\repo\.venv\Lib\site-packages\sklearn\utils
\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all_fi
nite' in 1.6 and will be removed in 1.8.
  warnings.warn(
```




```
Number of Components:  5 Evaluating Metric:  euclidean
Homogeneity : 0.492
Completeness : 0.520
V-measure : 0.506
Adjusted Rand-Index : 0.393
Adjusted Mutual Information Score : 0.504
c:\Users\astro\repositories\UCLA\ECE 219\repo\.venv\Lib\site-packages\sklearn\utils
\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all_fi
nite' in 1.6 and will be removed in 1.8.
  warnings.warn(
```

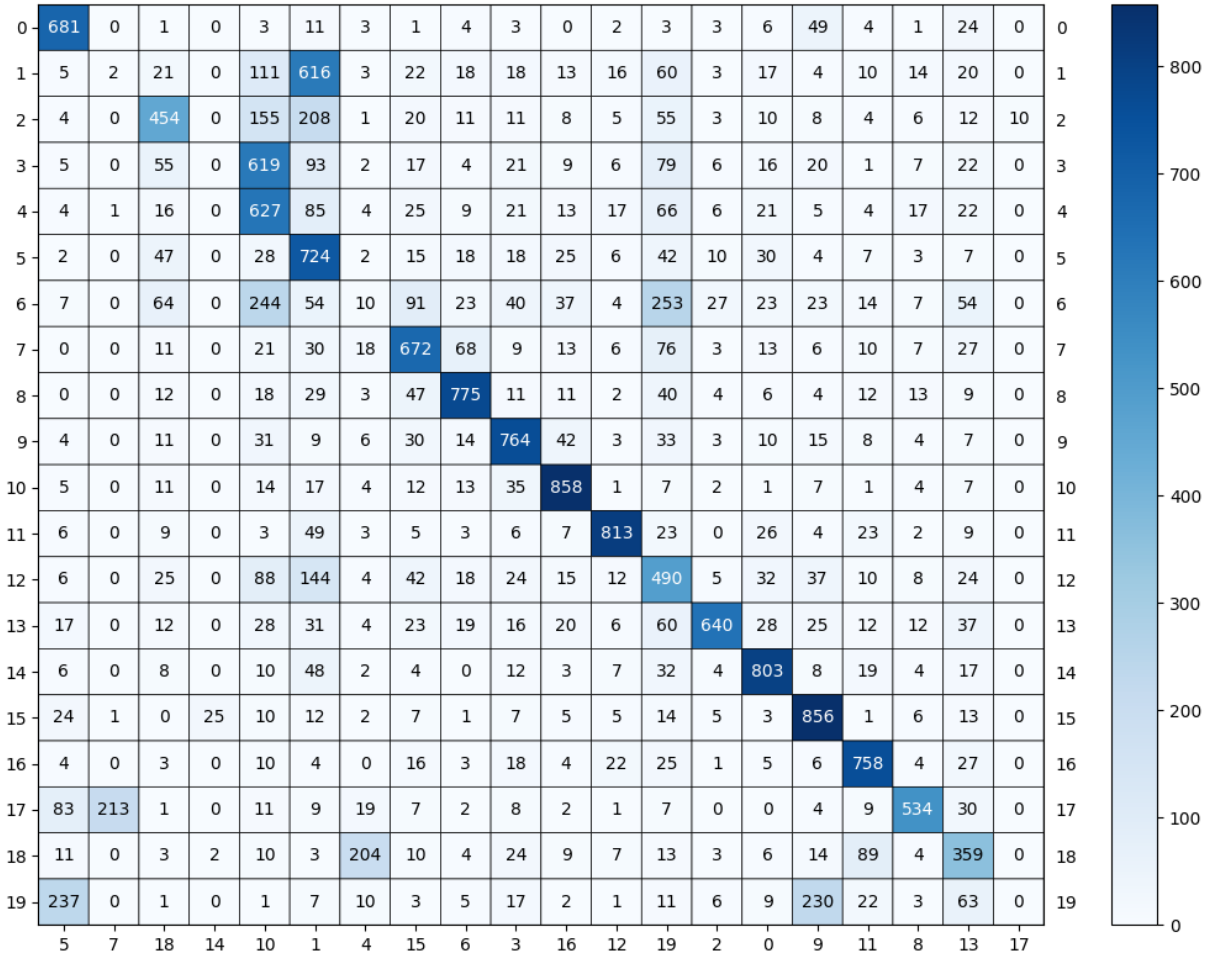



```
Number of Components:  20 Evaluating Metric:  cosine
Homogeneity : 0.482
Completeness : 0.532
V-measure : 0.506
Adjusted Rand-Index : 0.368
Adjusted Mutual Information Score : 0.504
c:\Users\astro\repositories\UCLA\ECE 219\repo\.venv\Lib\site-packages\sklearn\utils
\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all_fi
nite' in 1.6 and will be removed in 1.8.
  warnings.warn(
```



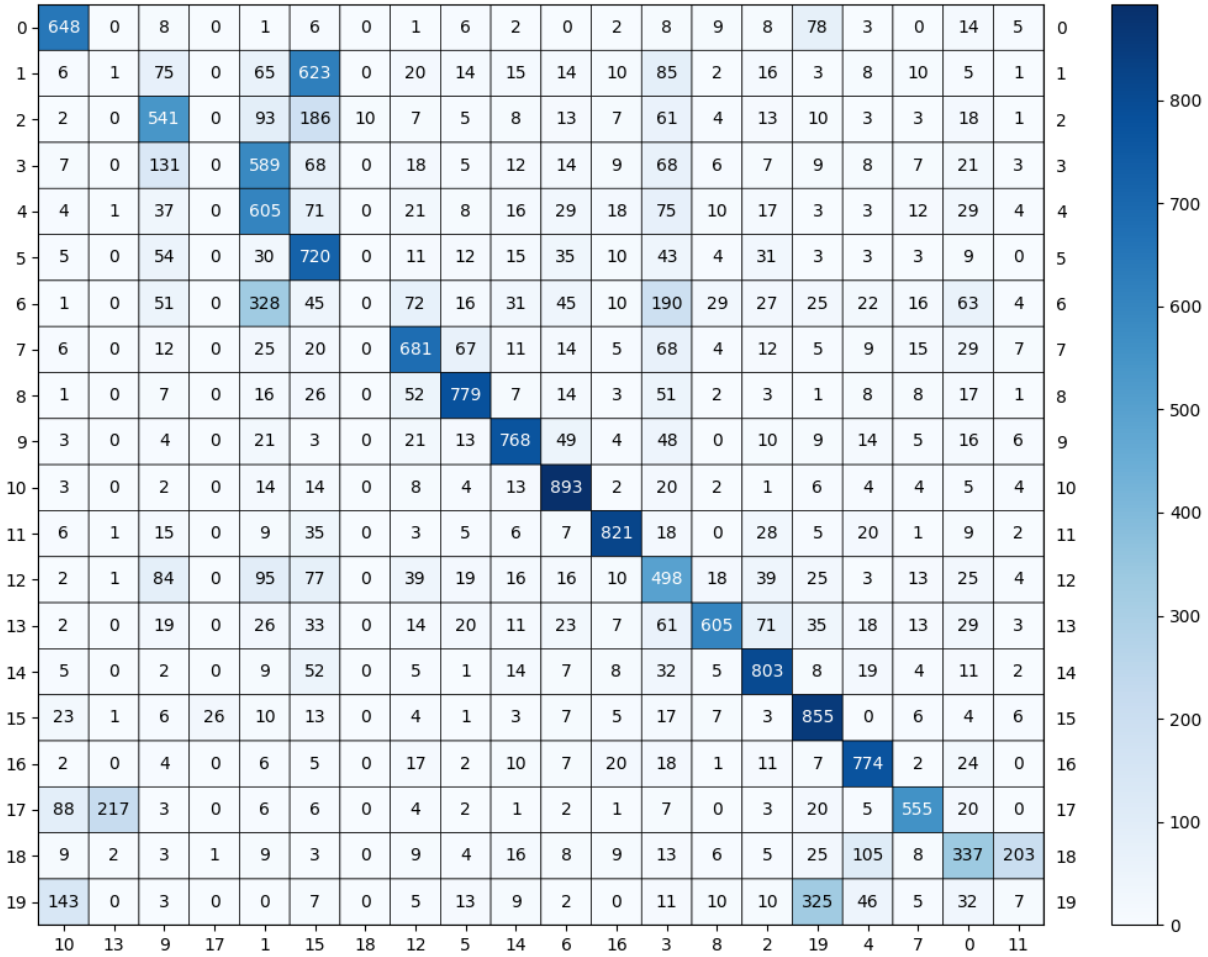
Number of Components: 20 Evaluating Metric: euclidean
Homogeneity : 0.499
Completeness : 0.529
V-measure : 0.514
Adjusted Rand-Index : 0.403
Adjusted Mutual Information Score : 0.512

```
c:\Users\astro\repositories\UCLA\ECE 219\repo\.venv\Lib\site-packages\sklearn\utils
\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all_fi
nite' in 1.6 and will be removed in 1.8.
  warnings.warn(
```



Number of Components: 200 Evaluating Metric: cosine
Homogeneity : 0.501
Completeness : 0.536
V-measure : 0.518
Adjusted Rand-Index : 0.396
Adjusted Mutual Information Score : 0.516

```
c:\Users\astro\repositories\UCLA\ECE 219\repo\.venv\Lib\site-packages\sklearn\utils
\deprecation.py:151: FutureWarning: 'force_all_finite' was renamed to 'ensure_all_fi
nite' in 1.6 and will be removed in 1.8.
  warnings.warn(
```




```
Number of Components:  200 Evaluating Metric:  euclidean
Homogeneity : 0.508
Completeness : 0.543
V-measure : 0.525
Adjusted Rand-Index : 0.405
Adjusted Mutual Information Score : 0.523
```

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

Based on our evaluations of the 6 combinations, the iteration with a number of components of 200 and the metric of euclidean, was able to achieve the highest scores on all 5 metrics when compared against its counter parts. Indicating that these setting lead to the most optimal combination of hyper parameters for the full newsgroup data.

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.

Based on the Contingency matrices created throughout the project document, we can see that for a K-means implementation with 20 classes, our best choices would likely be UMAP-

reduction, with 20 components, and a metric of euclidian, as this combination of hyperparameters will allow us to best represent the our dataset without lossing valuable information, or overfitting to our dataset.

QUESTION 14: Use UMAP to reduce the dimensionality properly, and perform Agglomerative clustering with n_clusters=20 . Compare the performance of “ward” and “single” linkage criteria.

Report the five clustering evaluation metrics for each case.

In [78]:
```
umap_full = umap.UMAP(n_components=20, metric='euclidean').fit_transform(tfidf_pred
agg_cluster = AgglomerativeClustering(n_clusters = 20, linkage = 'ward')
agg_pred = agg_cluster.fit(umap_full)
y_pred = agg_pred.labels_

print("Homogeneity : ", homogeneity_score(data_full.target, y_pred))
print("Completeness : ",  completeness_score(data_full.target, y_pred))
print("V-measure : ",  v_measure_score(data_full.target, y_pred))
print("Adjusted Rand-Index : ", adjusted_rand_score(data_full.target, y_pred))
print("Adjusted Mutual Information Score : ", adjusted_mutual_info_score(data_full.

umap_full = umap.UMAP(n_components=20, metric='euclidean').fit_transform(tfidf_pred
agg_cluster = AgglomerativeClustering(n_clusters = 20, linkage = 'single')
y_pred = agg_cluster.fit_predict(umap_full)
# y_pred = agg_pred.labels_

print("Homogeneity : " , homogeneity_score(data_full.target, y_pred))
print("Completeness : " , completeness_score(data_full.target, y_pred))
print("V-measure : " , v_measure_score(data_full.target, y_pred))
print("Adjusted Rand-Index : ", adjusted_rand_score(data_full.target, y_pred))
print("Adjusted Mutual Information Score : ", adjusted_mutual_info_score(data_full.
```

```
Homogeneity :  0.47905936599646803
Completeness :  0.5289501549781318
V-measure :  0.5027701040810756
Adjusted Rand-Index :  0.35129580544475375
Adjusted Mutual Information Score :  0.5010551171337507
Homogeneity :  0.020057199898966618
Completeness :  0.3569396172033896
V-measure :  0.03798021059772106
Adjusted Rand-Index :  0.0005978820980958837
Adjusted Mutual Information Score :  0.03252735577005047
```

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.

In [77]:
```
cluster_sizes = [20, 100, 200]
```

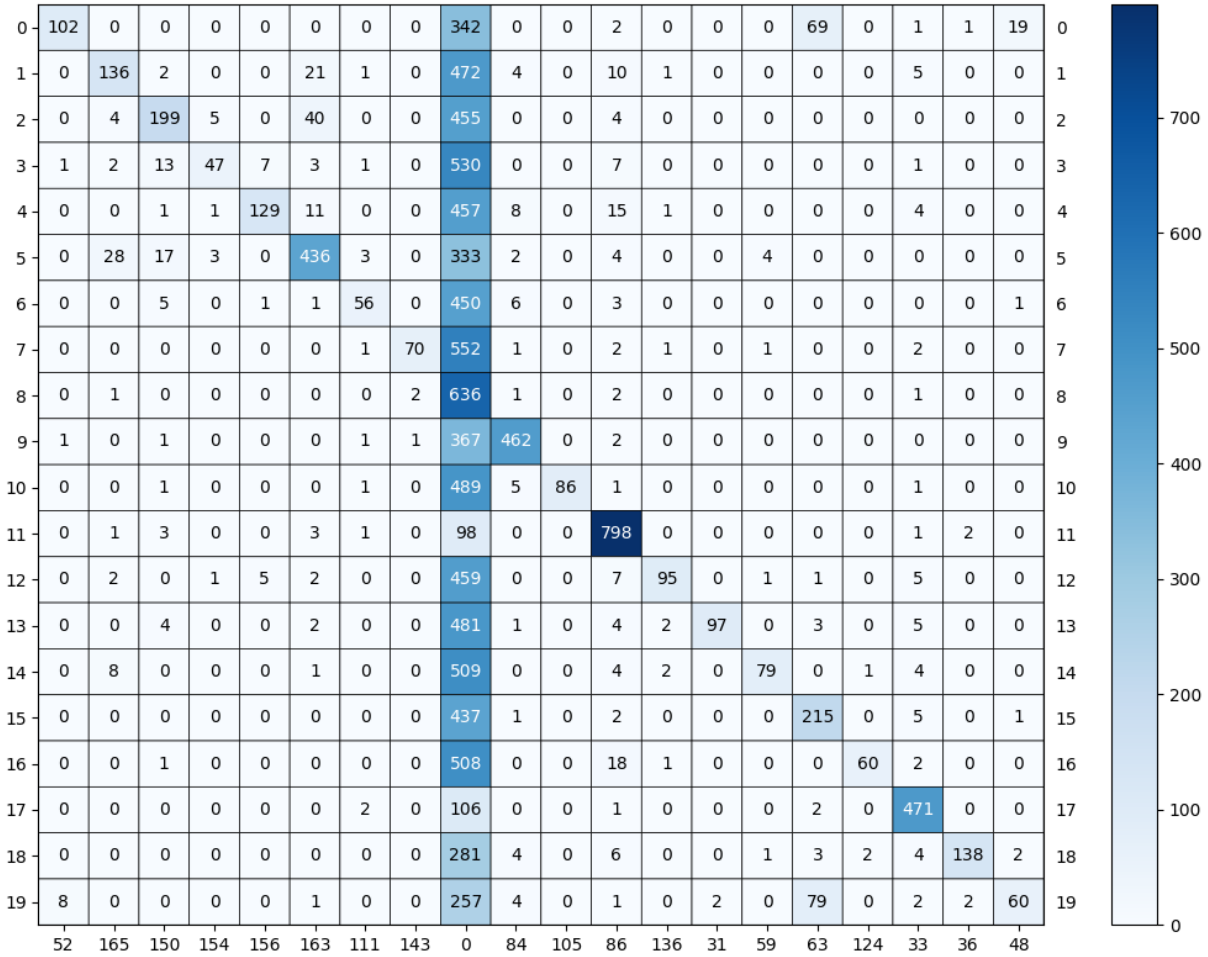
```
umap_data = umap.UMAP(n_components=20, metric='euclidean').fit_transform(tfidf_pred

for i in cluster_sizes:
    scan = hdbscan.HDBSCAN(min_cluster_size = i)
    y_pred = scan.fit_predict(umap_data)
    print("Cluser Size : ", i)
    print("Homogeneity : " , homogeneity_score(data_full.target, y_pred))
    print("Completeness : " , completeness_score(data_full.target, y_pred))
    print("V-measure : " , v_measure_score(data_full.target, y_pred))
    print("Adjusted Rand-Index : ", adjusted_rand_score(data_full.target, y_pred))
    print("Adjusted Mutual Information Score : ", adjusted_mutual_info_score(data_f
```

```
Cluser Size :  20
Homogeneity :  0.38487866199768805
Completeness :  0.3513118772083331
V-measure :  0.36733002678807974
Adjusted Rand-Index :  0.03203092211313788
Adjusted Mutual Information Score :  0.3453488956464685
Cluser Size :  100
Homogeneity :  0.013375875417440685
Completeness :  0.45514710358558597
V-measure :  0.025988014364309076
Adjusted Rand-Index :  0.000578056376781946
Adjusted Mutual Information Score :  0.025320405662514343
Cluser Size :  200
Homogeneity :  0.013375875417440685
Completeness :  0.45514710358558597
V-measure :  0.025988014364309076
Adjusted Rand-Index :  0.000578056376781946
Adjusted Mutual Information Score :  0.025320405662514343
```

QUESTION 16: Contingency matrix 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.

In [79]:
```
umap_data = umap.UMAP(n_components=20, metric='euclidean').fit_transform(tfidf_pred
scan = hdbscan.HDBSCAN(min_cluster_size = 20)
y_pred = scan.fit_predict(umap_data)

cm = contingency_matrix(data_full.target, y_pred)
rows, cols = linear_sum_assignment(cm, maximize=True)
reordered_mat = cm[np.ix_(rows, cols)]
plotmat.plot_mat(reordered_mat, xticklabels=cols, yticklabels=rows, size=(10,8))
```



Based on the resulting contingency matrix created using HDBSCAN, we can see that the number of clusters is 20, which was the best from our results in question 15. According to HDBSCAN documentation, -1 signifies a cluster that has been classified by the model as noise and do not belong to any of the identified clusters.

QUESTION 17: 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.

Hint: DBSCAN and HDBSCAN do not accept the number of clusters as an input parameter. So pay close attention to how the different clustering metrics are being computed for these methods.

In [82]:
```
def compare_models(models_metrics):
    best_model = None
    best_score = -float('inf')
    for model, metrics in models_metrics.items():
        cumulative_score = np.sum(metrics)
        if cumulative_score > best_score:
            best_score = cumulative_score
            best_model = model
```

```
    return best_model, best_score
```

In [ ]:

```
# Takes 1 hour and 12 minutes to fully run
dimentionallity_reduction = [TruncatedSVD(n_components=5, random_state=42),
                             TruncatedSVD(n_components=20, random_state=42),
                             TruncatedSVD(n_components=200, random_state=42),
                             NMF(n_components=5, random_state=42),
                             NMF(n_components=20, random_state=42),
                             NMF(n_components=200, random_state=42),
                             umap.UMAP(n_components=5, metric='euclidean'),
                             umap.UMAP(n_components=20, metric='euclidean'),
                             umap.UMAP(n_components=200, metric='euclidean')]

cluster_techniques = [KMeans(n_clusters=10, random_state=42, max_iter=1000, n_init=
                      KMeans(n_clusters=20, random_state=42, max_iter=1000, n_init=
                      KMeans(n_clusters=50, random_state=42, max_iter=1000, n_init=
                      AgglomerativeClustering(n_clusters=20),
                      hdbscan.HDBSCAN(min_cluster_size=100),
                      hdbscan.HDBSCAN(min_cluster_size=200)]

model_metrics = {}
for i in dimentionallity_reduction:
    for c in cluster_techniques:
        if i is None:
            y_pred = c.fit_predict(tfidf_preds)
            hm_score = homogeneity_score(data_full.target, y_pred)
            comp_score = completeness_score(data_full.target, y_pred)
            v_score = v_measure_score(data_full.target, y_pred)
            rand_score = adjusted_rand_score(data_full.target, y_pred)
            mut_score = adjusted_mutual_info_score(data_full.target, y_pred)
            metrics_array = [hm_score, comp_score, v_score, rand_score, mut_score]
            model_metrics[f"{i},{c}"] = metrics_array
            print("Dimentionallity Reduction Type:", i, ", Clustering Type: ", c)
            print("Homogeneity : " , hm_score)
            print("Completeness : " , comp_score)
            print("V-measure : " , v_score)
            print("Adjusted Rand-Index : ", rand_score)
            print("Adjusted Mutual Information Score : ", mut_score)
        else:
            method = i.fit_transform(tfidf_preds)
            y_pred = c.fit_predict(method)
            hm_score = homogeneity_score(data_full.target, y_pred)
            comp_score = completeness_score(data_full.target, y_pred)
            v_score = v_measure_score(data_full.target, y_pred)
            rand_score = adjusted_rand_score(data_full.target, y_pred)
            mut_score = adjusted_mutual_info_score(data_full.target, y_pred)
            metrics_array = [hm_score, comp_score, v_score, rand_score, mut_score]
            model_metrics[f"{i},{c}"] = metrics_array
            print("Dimentionallity Reduction Type:", i, ", Clustering Type: ", c)
            print("Homogeneity : " , hm_score)
            print("Completeness : " , comp_score)
            print("V-measure : " , v_score)
            print("Adjusted Rand-Index : ", rand_score)
            print("Adjusted Mutual Information Score : ", mut_score)

best_model, best_score = compare_models(model_metrics)
```

```
print("Best Model Found is ", best_model)
print("With a score of ", best_score)
```

```
Dimentionallity Reduction Type: TruncatedSVD(n_components=5, random_state=42) , Clus
tering Type:  KMeans(max_iter=1000, n_clusters=10, n_init=30, random_state=42)
Homogeneity :  0.25273865828769104
Completeness :  0.3639992217811067
V-measure :  0.2983331425028029
Adjusted Rand-Index :  0.11262133015505373
Adjusted Mutual Information Score :  0.2970693666782106
Dimentionallity Reduction Type: TruncatedSVD(n_components=5, random_state=42) , Clus
tering Type:  KMeans(max_iter=1000, n_clusters=20, n_init=30, random_state=42)
Homogeneity :  0.3126207568897337
Completeness :  0.33144480930213405
V-measure :  0.32175769856430775
Adjusted Rand-Index :  0.12248763485114002
Adjusted Mutual Information Score :  0.3194995928145638
Dimentionallity Reduction Type: TruncatedSVD(n_components=5, random_state=42) , Clus
tering Type:  KMeans(max_iter=1000, n_clusters=50, n_init=30, random_state=42)
Homogeneity :  0.3617757672962524
Completeness :  0.29042847974142433
V-measure :  0.3221996378599729
Adjusted Rand-Index :  0.09263520495490955
Adjusted Mutual Information Score :  0.31708461648668484
Dimentionallity Reduction Type: TruncatedSVD(n_components=5, random_state=42) , Clus
tering Type:  AgglomerativeClustering(n_clusters=20)
Homogeneity :  0.30310687128330555
Completeness :  0.32738252149182245
V-measure :  0.31477735530316253
Adjusted Rand-Index :  0.12005558568374378
Adjusted Mutual Information Score :  0.3124748822850994
Dimentionallity Reduction Type: TruncatedSVD(n_components=5, random_state=42) , Clus
tering Type:  HDBSCAN(min_cluster_size=100)
Homogeneity :  0.0
Completeness :  1.0
V-measure :  0.0
Adjusted Rand-Index :  0.0
Adjusted Mutual Information Score :  0.0
Dimentionallity Reduction Type: TruncatedSVD(n_components=5, random_state=42) , Clus
tering Type:  HDBSCAN(min_cluster_size=200)
Homogeneity :  0.0
Completeness :  1.0
V-measure :  0.0
Adjusted Rand-Index :  0.0
Adjusted Mutual Information Score :  0.0
Dimentionallity Reduction Type: TruncatedSVD(n_components=20, random_state=42) , Clu
stering Type:  KMeans(max_iter=1000, n_clusters=10, n_init=30, random_state=42)
Homogeneity :  0.2761589150533012
Completeness :  0.43903515927729825
V-measure :  0.3390505531516107
Adjusted Rand-Index :  0.10122841884333234
Adjusted Mutual Information Score :  0.33780371066329795
Dimentionallity Reduction Type: TruncatedSVD(n_components=20, random_state=42) , Clu
stering Type:  KMeans(max_iter=1000, n_clusters=20, n_init=30, random_state=42)
Homogeneity :  0.2880305059891321
Completeness :  0.38347176056805204
V-measure :  0.32896855522245194
Adjusted Rand-Index :  0.09297012622849953
Adjusted Mutual Information Score :  0.3264581794062097
```

```
Dimentionallity Reduction Type: TruncatedSVD(n_components=20, random_state=42) , Clu
stering Type:  KMeans(max_iter=1000, n_clusters=50, n_init=30, random_state=42)
Homogeneity :  0.39611636948769197
Completeness :  0.3383394677762406
V-measure :  0.3649553719368441
Adjusted Rand-Index :  0.1384609662307626
Adjusted Mutual Information Score :  0.3599411973709435
Dimentionallity Reduction Type: TruncatedSVD(n_components=20, random_state=42) , Clu
stering Type:  AgglomerativeClustering(n_clusters=20)
Homogeneity :  0.33040464820347853
Completeness :  0.4333175252151034
V-measure :  0.37492724307908576
Adjusted Rand-Index :  0.12413059838732861
Adjusted Mutual Information Score :  0.37260624076173743
Dimentionallity Reduction Type: TruncatedSVD(n_components=20, random_state=42) , Clu
stering Type:  HDBSCAN(min_cluster_size=100)
Homogeneity :  0.0
Completeness :  1.0
V-measure :  0.0
Adjusted Rand-Index :  0.0
Adjusted Mutual Information Score :  0.0
Dimentionallity Reduction Type: TruncatedSVD(n_components=20, random_state=42) , Clu
stering Type:  HDBSCAN(min_cluster_size=200)
Homogeneity :  0.0
Completeness :  1.0
V-measure :  0.0
Adjusted Rand-Index :  0.0
Adjusted Mutual Information Score :  0.0
Dimentionallity Reduction Type: TruncatedSVD(n_components=200, random_state=42) , Cl
ustering Type:  KMeans(max_iter=1000, n_clusters=10, n_init=30, random_state=42)
Homogeneity :  0.2609774347277058
Completeness :  0.5093927418070324
V-measure :  0.3451328077722315
Adjusted Rand-Index :  0.0771553849700257
Adjusted Mutual Information Score :  0.34380329298014695
Dimentionallity Reduction Type: TruncatedSVD(n_components=200, random_state=42) , Cl
ustering Type:  KMeans(max_iter=1000, n_clusters=20, n_init=30, random_state=42)
Homogeneity :  0.29983314914549714
Completeness :  0.4371374610581335
V-measure :  0.35569478550118055
Adjusted Rand-Index :  0.0950019331687476
Adjusted Mutual Information Score :  0.3531683559256488
Dimentionallity Reduction Type: TruncatedSVD(n_components=200, random_state=42) , Cl
ustering Type:  KMeans(max_iter=1000, n_clusters=50, n_init=30, random_state=42)
Homogeneity :  0.33974086615180077
Completeness :  0.33392134864845885
V-measure :  0.3368059710757022
Adjusted Rand-Index :  0.09536142900515182
Adjusted Mutual Information Score :  0.3310842014816762
Dimentionallity Reduction Type: TruncatedSVD(n_components=200, random_state=42) , Cl
ustering Type:  AgglomerativeClustering(n_clusters=20)
Homogeneity :  0.22503950308985432
Completeness :  0.40902480948326814
V-measure :  0.2903388128059071
Adjusted Rand-Index :  0.04863740427366599
Adjusted Mutual Information Score :  0.2872824945873852
```

```
Dimentionallity Reduction Type: TruncatedSVD(n_components=200, random_state=42) , Cl
ustering Type:  HDBSCAN(min_cluster_size=100)
Homogeneity :  0.0
Completeness :  1.0
V-measure :  0.0
Adjusted Rand-Index :  0.0
Adjusted Mutual Information Score :  0.0
Dimentionallity Reduction Type: TruncatedSVD(n_components=200, random_state=42) , Cl
ustering Type:  HDBSCAN(min_cluster_size=200)
Homogeneity :  0.0
Completeness :  1.0
V-measure :  0.0
Adjusted Rand-Index :  0.0
Adjusted Mutual Information Score :  0.0
Dimentionallity Reduction Type: NMF(n_components=5, random_state=42) , Clustering Ty
pe:  KMeans(max_iter=1000, n_clusters=10, n_init=30, random_state=42)
Homogeneity :  0.25649428568550203
Completeness :  0.3676581095715578
V-measure :  0.30217685586931053
Adjusted Rand-Index :  0.10408478491728526
Adjusted Mutual Information Score :  0.3009228753354001
Dimentionallity Reduction Type: NMF(n_components=5, random_state=42) , Clustering Ty
pe:  KMeans(max_iter=1000, n_clusters=20, n_init=30, random_state=42)
Homogeneity :  0.2931683187383885
Completeness :  0.3101435167296279
V-measure :  0.30141710479358386
Adjusted Rand-Index :  0.10550265570268441
Adjusted Mutual Information Score :  0.2990926697960212
Dimentionallity Reduction Type: NMF(n_components=5, random_state=42) , Clustering Ty
pe:  KMeans(max_iter=1000, n_clusters=50, n_init=30, random_state=42)
Homogeneity :  0.3299195953342298
Completeness :  0.2627904885764448
V-measure :  0.29255359070925285
Adjusted Rand-Index :  0.07647432708367888
Adjusted Mutual Information Score :  0.28723456699452027
Dimentionallity Reduction Type: NMF(n_components=5, random_state=42) , Clustering Ty
pe:  AgglomerativeClustering(n_clusters=20)
Homogeneity :  0.2872534363748666
Completeness :  0.30222936082300966
V-measure :  0.2945511654707259
Adjusted Rand-Index :  0.11037234598060279
Adjusted Mutual Information Score :  0.2922115256442744
Dimentionallity Reduction Type: NMF(n_components=5, random_state=42) , Clustering Ty
pe:  HDBSCAN(min_cluster_size=100)
Homogeneity :  0.018042395354984052
Completeness :  0.14975151256474453
V-measure :  0.03220469715733114
Adjusted Rand-Index :  0.0008352298017490346
Adjusted Mutual Information Score :  0.03161902552279655
Dimentionallity Reduction Type: NMF(n_components=5, random_state=42) , Clustering Ty
pe:  HDBSCAN(min_cluster_size=200)
Homogeneity :  0.011350452851269456
Completeness :  0.12530464512554135
V-measure :  0.02081538834041675
Adjusted Rand-Index :  0.00022356061900455452
Adjusted Mutual Information Score :  0.02020529992144077
```

```
Dimentionallity Reduction Type: NMF(n_components=20, random_state=42) , Clustering T
ype:  KMeans(max_iter=1000, n_clusters=10, n_init=30, random_state=42)
Homogeneity :  0.16871361838082372
Completeness :  0.5362928185031597
V-measure :  0.25667822926901274
Adjusted Rand-Index :  0.025189383124498123
Adjusted Mutual Information Score :  0.2549250310861421
Dimentionallity Reduction Type: NMF(n_components=20, random_state=42) , Clustering T
ype:  KMeans(max_iter=1000, n_clusters=20, n_init=30, random_state=42)
Homogeneity :  0.2746972019021395
Completeness :  0.3893870356552565
V-measure :  0.32213843696967087
Adjusted Rand-Index :  0.07128311820408853
Adjusted Mutual Information Score :  0.31953319755372406
Dimentionallity Reduction Type: NMF(n_components=20, random_state=42) , Clustering T
ype:  KMeans(max_iter=1000, n_clusters=50, n_init=30, random_state=42)
Homogeneity :  0.37196007906519885
Completeness :  0.3242952887551911
V-measure :  0.3464961473071744
Adjusted Rand-Index :  0.12201005235975028
Adjusted Mutual Information Score :  0.3412428467873473
Dimentionallity Reduction Type: NMF(n_components=20, random_state=42) , Clustering T
ype:  AgglomerativeClustering(n_clusters=20)
Homogeneity :  0.2768374244938308
Completeness :  0.3949356452900103
V-measure :  0.3255056560039718
Adjusted Rand-Index :  0.06410355909524232
Adjusted Mutual Information Score :  0.3229069646927765
Dimentionallity Reduction Type: NMF(n_components=20, random_state=42) , Clustering T
ype:  HDBSCAN(min_cluster_size=100)
Homogeneity :  0.006673916486519139
Completeness :  0.12738529922358485
V-measure :  0.012683333169229937
Adjusted Rand-Index :  3.1645109232318096e-05
Adjusted Mutual Information Score :  0.012038709480669832
Dimentionallity Reduction Type: NMF(n_components=20, random_state=42) , Clustering T
ype:  HDBSCAN(min_cluster_size=200)
Homogeneity :  0.0
Completeness :  1.0
V-measure :  0.0
Adjusted Rand-Index :  0.0
Adjusted Mutual Information Score :  0.0
Dimentionallity Reduction Type: NMF(n_components=200, random_state=42) , Clustering
Type:  KMeans(max_iter=1000, n_clusters=10, n_init=30, random_state=42)
Homogeneity :  0.0692691383830504
Completeness :  0.40107876168497913
V-measure :  0.1181354492776133
Adjusted Rand-Index :  0.00498138969246422
Adjusted Mutual Information Score :  0.11572861944688817
Dimentionallity Reduction Type: NMF(n_components=200, random_state=42) , Clustering
Type:  KMeans(max_iter=1000, n_clusters=20, n_init=30, random_state=42)
Homogeneity :  0.0968298545793245
Completeness :  0.3354783956939782
V-measure :  0.1502831567476059
Adjusted Rand-Index :  0.005886402472406172
Adjusted Mutual Information Score :  0.14580247457679077
```

```
Dimentionallity Reduction Type: NMF(n_components=200, random_state=42) , Clustering
Type:  KMeans(max_iter=1000, n_clusters=50, n_init=30, random_state=42)
Homogeneity :  0.16176739651248698
Completeness :  0.257727141573628
V-measure :  0.19877183094294462
Adjusted Rand-Index :  0.009077259642256795
Adjusted Mutual Information Score :  0.1899536138916994
Dimentionallity Reduction Type: NMF(n_components=200, random_state=42) , Clustering
Type:  AgglomerativeClustering(n_clusters=20)
Homogeneity :  0.1010158535752369
Completeness :  0.35472971837494793
V-measure :  0.1572514468404556
Adjusted Rand-Index :  0.007399304083035572
Adjusted Mutual Information Score :  0.15271962544680398
Dimentionallity Reduction Type: NMF(n_components=200, random_state=42) , Clustering
Type:  HDBSCAN(min_cluster_size=100)
Homogeneity :  0.0
Completeness :  1.0
V-measure :  0.0
Adjusted Rand-Index :  0.0
Adjusted Mutual Information Score :  0.0
Dimentionallity Reduction Type: NMF(n_components=200, random_state=42) , Clustering
Type:  HDBSCAN(min_cluster_size=200)
Homogeneity :  0.0
Completeness :  1.0
V-measure :  0.0
Adjusted Rand-Index :  0.0
Adjusted Mutual Information Score :  0.0
Dimentionallity Reduction Type: UMAP(n_components=5, tqdm_kwds={'bar_format': '{des
c}: {percentage:3.0f}%| {bar} {n_fmt}/{total_fmt} [{elapsed}]', 'desc': 'Epochs comp
leted', 'disable': True}) , Clustering Type:  KMeans(max_iter=1000, n_clusters=10, n
_init=30, random_state=42)
Homogeneity :  0.3986837912544073
Completeness :  0.5843308322641636
V-measure :  0.4739771433309075
Adjusted Rand-Index :  0.2511056109711637
Adjusted Mutual Information Score :  0.4730223546143701
Dimentionallity Reduction Type: UMAP(n_components=5, tqdm_kwds={'bar_format': '{des
c}: {percentage:3.0f}%| {bar} {n_fmt}/{total_fmt} [{elapsed}]', 'desc': 'Epochs comp
leted', 'disable': True}) , Clustering Type:  KMeans(max_iter=1000, n_clusters=20, n
_init=30, random_state=42)
Homogeneity :  0.5074283036081111
Completeness :  0.5409278040799056
V-measure :  0.5236428270619873
Adjusted Rand-Index :  0.40588207428305056
Adjusted Mutual Information Score :  0.5220390593957115
Dimentionallity Reduction Type: UMAP(n_components=5, tqdm_kwds={'bar_format': '{des
c}: {percentage:3.0f}%| {bar} {n_fmt}/{total_fmt} [{elapsed}]', 'desc': 'Epochs comp
leted', 'disable': True}) , Clustering Type:  KMeans(max_iter=1000, n_clusters=50, n
_init=30, random_state=42)
Homogeneity :  0.5699559580140793
Completeness :  0.4578535804893018
V-measure :  0.5077913102031363
Adjusted Rand-Index :  0.3677076016137933
Adjusted Mutual Information Score :  0.5040620995506321
Dimentionallity Reduction Type: UMAP(n_components=5, tqdm_kwds={'bar_format': '{des
```

```
c}: {percentage:3.0f}%| {bar} {n_fmt}/{total_fmt} [{elapsed}]', 'desc': 'Epochs comp
leted', 'disable': True}) , Clustering Type:  AgglomerativeClustering(n_clusters=20)
Homogeneity :  0.4887317332825636
Completeness :  0.5146456621942461
V-measure :  0.5013540620795457
Adjusted Rand-Index :  0.37706892236663747
Adjusted Mutual Information Score :  0.49968597959687555
Dimentionallity Reduction Type: UMAP(n_components=5, tqdm_kwds={'bar_format': '{des
c}: {percentage:3.0f}%| {bar} {n_fmt}/{total_fmt} [{elapsed}]', 'desc': 'Epochs comp
leted', 'disable': True}) , Clustering Type:  HDBSCAN(min_cluster_size=100)
Homogeneity :  0.2717337961244501
Completeness :  0.5928219231236855
V-measure :  0.37265325533046806
Adjusted Rand-Index :  0.08277277156280796
Adjusted Mutual Information Score :  0.371482047092798
Dimentionallity Reduction Type: UMAP(n_components=5, tqdm_kwds={'bar_format': '{des
c}: {percentage:3.0f}%| {bar} {n_fmt}/{total_fmt} [{elapsed}]', 'desc': 'Epochs comp
leted', 'disable': True}) , Clustering Type:  HDBSCAN(min_cluster_size=200)
Homogeneity :  0.013906166787540425
Completeness :  0.49490187459805834
V-measure :  0.027052198282422068
Adjusted Rand-Index :  0.0005964497395942732
Adjusted Mutual Information Score :  0.026369565026625134
Dimentionallity Reduction Type: UMAP(n_components=20, tqdm_kwds={'bar_format': '{des
c}: {percentage:3.0f}%| {bar} {n_fmt}/{total_fmt} [{elapsed}]', 'desc': 'Epochs comp
leted', 'disable': True}) , Clustering Type:  KMeans(max_iter=1000, n_clusters=10, n
_init=30, random_state=42)
Homogeneity :  0.39445037877495115
Completeness :  0.5780252323140412
V-measure :  0.46891103330073736
Adjusted Rand-Index :  0.2493576883355176
Adjusted Mutual Information Score :  0.4679471173468865
Dimentionallity Reduction Type: UMAP(n_components=20, tqdm_kwds={'bar_format': '{des
c}: {percentage:3.0f}%| {bar} {n_fmt}/{total_fmt} [{elapsed}]', 'desc': 'Epochs comp
leted', 'disable': True}) , Clustering Type:  KMeans(max_iter=1000, n_clusters=20, n
_init=30, random_state=42)
Homogeneity :  0.5119337513721676
Completeness :  0.5381320725495577
V-measure :  0.5247060981474436
Adjusted Rand-Index :  0.41551890463133057
Adjusted Mutual Information Score :  0.5231175542682646
Dimentionallity Reduction Type: UMAP(n_components=20, tqdm_kwds={'bar_format': '{des
c}: {percentage:3.0f}%| {bar} {n_fmt}/{total_fmt} [{elapsed}]', 'desc': 'Epochs comp
leted', 'disable': True}) , Clustering Type:  KMeans(max_iter=1000, n_clusters=50, n
_init=30, random_state=42)
Homogeneity :  0.5750284037922084
Completeness :  0.47269349466591215
V-measure :  0.5188632329260514
Adjusted Rand-Index :  0.41294533327743965
Adjusted Mutual Information Score :  0.515143073844444
Dimentionallity Reduction Type: UMAP(n_components=20, tqdm_kwds={'bar_format': '{des
c}: {percentage:3.0f}%| {bar} {n_fmt}/{total_fmt} [{elapsed}]', 'desc': 'Epochs comp
leted', 'disable': True}) , Clustering Type:  AgglomerativeClustering(n_clusters=20)
Homogeneity :  0.4911007246827806
Completeness :  0.5289822658448694
V-measure :  0.5093381156495482
```

```
Adjusted Rand-Index :  0.37191898467480083
Adjusted Mutual Information Score :  0.5076773718257355
Dimentionallity Reduction Type: UMAP(n_components=20, tqdm_kwds={'bar_format': '{des
c}: {percentage:3.0f}%| {bar} {n_fmt}/{total_fmt} [{elapsed}]', 'desc': 'Epochs comp
leted', 'disable': True}) , Clustering Type:  HDBSCAN(min_cluster_size=100)
Homogeneity :  0.014010022581543269
Completeness :  0.4885939357545715
V-measure :  0.02723898990285681
Adjusted Rand-Index :  0.0006287039623015205
Adjusted Mutual Information Score :  0.026566033073556997
Dimentionallity Reduction Type: UMAP(n_components=20, tqdm_kwds={'bar_format': '{des
c}: {percentage:3.0f}%| {bar} {n_fmt}/{total_fmt} [{elapsed}]', 'desc': 'Epochs comp
leted', 'disable': True}) , Clustering Type:  HDBSCAN(min_cluster_size=200)
Homogeneity :  0.013557469049912231
Completeness :  0.48286758874085706
V-measure :  0.026374423638861062
Adjusted Rand-Index :  0.0005986558796979328
Adjusted Mutual Information Score :  0.025699141711480303
Dimentionallity Reduction Type: UMAP(n_components=200, tqdm_kwds={'bar_format': '{de
sc}: {percentage:3.0f}%| {bar} {n_fmt}/{total_fmt} [{elapsed}]', 'desc': 'Epochs com
pleted', 'disable': True}) , Clustering Type:  KMeans(max_iter=1000, n_clusters=10,
n_init=30, random_state=42)
Homogeneity :  0.395120049919619
Completeness :  0.5833859311223732
V-measure :  0.47114168475907026
Adjusted Rand-Index :  0.24281841749217073
Adjusted Mutual Information Score :  0.4701790519945202
Dimentionallity Reduction Type: UMAP(n_components=200, tqdm_kwds={'bar_format': '{de
sc}: {percentage:3.0f}%| {bar} {n_fmt}/{total_fmt} [{elapsed}]', 'desc': 'Epochs com
pleted', 'disable': True}) , Clustering Type:  KMeans(max_iter=1000, n_clusters=20,
n_init=30, random_state=42)
Homogeneity :  0.5073754024789282
Completeness :  0.5431840793698351
V-measure :  0.5246694654650517
Adjusted Rand-Index :  0.4014722822240333
Adjusted Mutual Information Score :  0.5230658817578807
Dimentionallity Reduction Type: UMAP(n_components=200, tqdm_kwds={'bar_format': '{de
sc}: {percentage:3.0f}%| {bar} {n_fmt}/{total_fmt} [{elapsed}]', 'desc': 'Epochs com
pleted', 'disable': True}) , Clustering Type:  KMeans(max_iter=1000, n_clusters=50,
n_init=30, random_state=42)
Homogeneity :  0.566504621931102
Completeness :  0.4597641592492361
V-measure :  0.5075834440045996
Adjusted Rand-Index :  0.404035056610579
Adjusted Mutual Information Score :  0.5038233860995058
Dimentionallity Reduction Type: UMAP(n_components=200, tqdm_kwds={'bar_format': '{de
sc}: {percentage:3.0f}%| {bar} {n_fmt}/{total_fmt} [{elapsed}]', 'desc': 'Epochs com
pleted', 'disable': True}) , Clustering Type:  AgglomerativeClustering(n_clusters=2
0)
Homogeneity :  0.48458902760676054
Completeness :  0.5390625372998407
V-measure :  0.5103763814266179
Adjusted Rand-Index :  0.3539967064684246
Adjusted Mutual Information Score :  0.5086687465481179
Dimentionallity Reduction Type: UMAP(n_components=200, tqdm_kwds={'bar_format': '{de
sc}: {percentage:3.0f}%| {bar} {n_fmt}/{total_fmt} [{elapsed}]', 'desc': 'Epochs com
```

```
pleted', 'disable': True}) , Clustering Type:  HDBSCAN(min_cluster_size=100)
Homogeneity :  0.01394077175109458
Completeness :  0.4825762697636899
V-measure :  0.02709870988011164
Adjusted Rand-Index :  0.0006384068573917776
Adjusted Mutual Information Score :  0.026425332954461
Dimentionallity Reduction Type: UMAP(n_components=200, tqdm_kwds={'bar_format': '{de
sc}: {percentage:3.0f}%| {bar} {n_fmt}/{total_fmt} [{elapsed}]', 'desc': 'Epochs com
pleted', 'disable': True}) , Clustering Type:  HDBSCAN(min_cluster_size=200)
Homogeneity :  0.013754781970847894
Completeness :  0.45727709920099674
V-measure :  0.026706246652025794
Adjusted Rand-Index :  0.0006068234788160461
Adjusted Mutual Information Score :  0.026039459057344345
Best Model Found is  UMAP(n_components=20, tqdm_kwds={'bar_format': '{desc}: {percen
tage:3.0f}%| {bar} {n_fmt}/{total_fmt} [{elapsed}]', 'desc': 'Epochs completed', 'di
sable': True}),KMeans(max_iter=1000, n_clusters=20, n_init=30, random_state=42)
With a score of  2.513408380968764
```

We can safely assume that without dimentionality reduction, we will have a suboptimal clustering result, furthermore, from our experiments, we can see that of the available techniques, a UMAP with 20 components and Kmeans and 20 clusters preforms best with the following metrics:

- Homogeneity : 0.5119337513721676
- Completeness : 0.5381320725495577
- V-measure : 0.5247060981474436
- Adjusted Rand-Index : 0.41551890463133057
- Adjusted Mutual Information Score : 0.5231175542682646

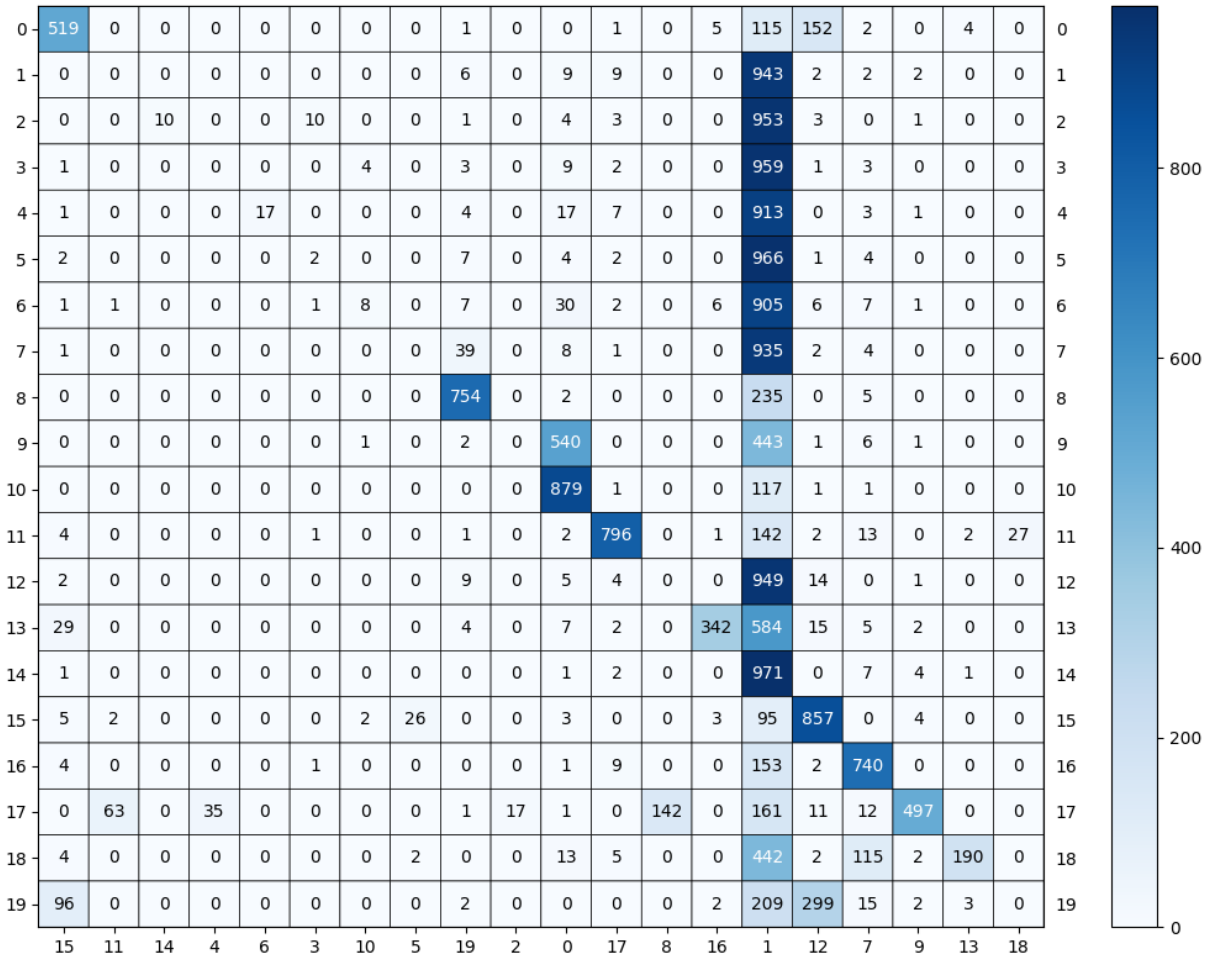
QUESTION 18: Extra credit: If you can find creative ways to further enhance the clustering performance, report your method and the results you obtain.

In [103…

```python
from sklearn.cluster import SpectralClustering
spectral = SpectralClustering(n_clusters=20, affinity='nearest_neighbors')
y_pred = spectral.fit_predict(tfidf_preds)

cm = contingency_matrix(data_full.target, y_pred)

hm_score = homogeneity_score(data_full.target, y_pred)
comp_score = completeness_score(data_full.target, y_pred)
v_score = v_measure_score(data_full.target, y_pred)
rand_score = adjusted_rand_score(data_full.target, y_pred)
mut_score = adjusted_mutual_info_score(data_full.target, y_pred)

print("Homogeneity : " , hm_score)
print("Completeness : " , comp_score)
print("V-measure : " , v_score)
print("Adjusted Rand-Index : ", rand_score)
print("Adjusted Mutual Information Score : ", mut_score)

rows, cols = linear_sum_assignment(cm, maximize=True)
reordered_mat = cm[np.ix_(rows, cols)]
```

```
plotmat.plot_mat(reordered_mat, xticklabels=cols, yticklabels=rows, size=(10,8))
```

```
Homogeneity :  0.35324846286571226
Completeness :  0.6647802762512584
V-measure :  0.46134770405969555
Adjusted Rand-Index :  0.09668461226948066
Adjusted Mutual Information Score :  0.4589353981944944
```



Unfortunatly, we were not able to get better results than our UMAP and KMeans implementation from part 17, however, some possible suggestions would be playing with UMAP hyperparameters like the number of neighbors, minimum distance from the center of the cluster, or attempting different embedding strategies to find the most optimal configuration.