Part 1

Clustering with Sparse Text Representations

In [ ]:
```
!pip install regex
!pip install nltk
!pip install sklearn
!pip install umap-learn[plot]
!pip install holoviews
!pip install -U ipykernel
!pip install ClusterEnsembles
```

```
Requirement already satisfied: regex in /usr/local/lib/python3.10/dist-packages
(2023.6.3)
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages
(3.8.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages
(from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages
(from nltk) (1.3.2)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-
packages (from nltk) (2023.6.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (f
rom nltk) (4.66.1)
Collecting sklearn
  Using cached sklearn-0.0.post12.tar.gz (2.6 kB)
  error: subprocess-exited-with-error

  × python setup.py egg_info did not run successfully.
  │ exit code: 1
  ╰─> See above for output.

  note: This error originates from a subprocess, and is likely not a problem with
pip.
  Preparing metadata (setup.py) ... error
error: metadata-generation-failed

× Encountered error while generating package metadata.
╰─> See above for output.

note: This is an issue with the package mentioned above, not pip.
hint: See above for details.
Requirement already satisfied: umap-learn[plot] in /usr/local/lib/python3.10/dist
-packages (0.5.5)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-pack
ages (from umap-learn[plot]) (1.23.5)
Requirement already satisfied: scipy>=1.3.1 in /usr/local/lib/python3.10/dist-pac
kages (from umap-learn[plot]) (1.11.4)
Requirement already satisfied: scikit-learn>=0.22 in /usr/local/lib/python3.10/di
st-packages (from umap-learn[plot]) (1.2.2)
Requirement already satisfied: numba>=0.51.2 in /usr/local/lib/python3.10/dist-pa
ckages (from umap-learn[plot]) (0.58.1)
Requirement already satisfied: pynndescent>=0.5 in /usr/local/lib/python3.10/dist
-packages (from umap-learn[plot]) (0.5.11)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (f
rom umap-learn[plot]) (4.66.1)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages
(from umap-learn[plot]) (1.5.3)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packa
ges (from umap-learn[plot]) (3.7.1)
Requirement already satisfied: datashader in /usr/local/lib/python3.10/dist-packa
ges (from umap-learn[plot]) (0.16.0)
Requirement already satisfied: bokeh in /usr/local/lib/python3.10/dist-packages
(from umap-learn[plot]) (3.3.4)
Requirement already satisfied: holoviews in /usr/local/lib/python3.10/dist-packag
es (from umap-learn[plot]) (1.17.1)
Requirement already satisfied: colorcet in /usr/local/lib/python3.10/dist-package
s (from umap-learn[plot]) (3.0.1)
Requirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-packages
(from umap-learn[plot]) (0.13.1)
Requirement already satisfied: scikit-image in /usr/local/lib/python3.10/dist-pac
kages (from umap-learn[plot]) (0.19.3)
```

```
Requirement already satisfied: llvmlite<0.42,>=0.41.0dev0 in /usr/local/lib/pytho
n3.10/dist-packages (from numba>=0.51.2->umap-learn[plot]) (0.41.1)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.10/dist-pac
kages (from pynndescent>=0.5->umap-learn[plot]) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/
dist-packages (from scikit-learn>=0.22->umap-learn[plot]) (3.2.0)
Requirement already satisfied: Jinja2>=2.9 in /usr/local/lib/python3.10/dist-pack
ages (from bokeh->umap-learn[plot]) (3.1.3)
Requirement already satisfied: contourpy>=1 in /usr/local/lib/python3.10/dist-pac
kages (from bokeh->umap-learn[plot]) (1.2.0)
Requirement already satisfied: packaging>=16.8 in /usr/local/lib/python3.10/dist-
packages (from bokeh->umap-learn[plot]) (23.2)
Requirement already satisfied: pillow>=7.1.0 in /usr/local/lib/python3.10/dist-pa
ckages (from bokeh->umap-learn[plot]) (9.4.0)
Requirement already satisfied: PyYAML>=3.10 in /usr/local/lib/python3.10/dist-pac
kages (from bokeh->umap-learn[plot]) (6.0.1)
Requirement already satisfied: tornado>=5.1 in /usr/local/lib/python3.10/dist-pac
kages (from bokeh->umap-learn[plot]) (6.3.2)
Requirement already satisfied: xyzservices>=2021.09.1 in /usr/local/lib/python3.1
0/dist-packages (from bokeh->umap-learn[plot]) (2023.10.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.1
0/dist-packages (from pandas->umap-learn[plot]) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-pac
kages (from pandas->umap-learn[plot]) (2023.3.post1)
Requirement already satisfied: pyct>=0.4.4 in /usr/local/lib/python3.10/dist-pack
ages (from colorcet->umap-learn[plot]) (0.5.0)
Requirement already satisfied: dask in /usr/local/lib/python3.10/dist-packages (f
rom datashader->umap-learn[plot]) (2023.8.1)
Requirement already satisfied: multipledispatch in /usr/local/lib/python3.10/dist
-packages (from datashader->umap-learn[plot]) (1.0.0)
Requirement already satisfied: param in /usr/local/lib/python3.10/dist-packages
(from datashader->umap-learn[plot]) (2.0.2)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-package
s (from datashader->umap-learn[plot]) (2.31.0)
Requirement already satisfied: toolz in /usr/local/lib/python3.10/dist-packages
(from datashader->umap-learn[plot]) (0.12.1)
Requirement already satisfied: xarray in /usr/local/lib/python3.10/dist-packages
(from datashader->umap-learn[plot]) (2023.7.0)
Requirement already satisfied: pyviz-comms>=0.7.4 in /usr/local/lib/python3.10/di
st-packages (from holoviews->umap-learn[plot]) (3.0.1)
Requirement already satisfied: panel>=0.13.1 in /usr/local/lib/python3.10/dist-pa
ckages (from holoviews->umap-learn[plot]) (1.3.8)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-pac
kages (from matplotlib->umap-learn[plot]) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dis
t-packages (from matplotlib->umap-learn[plot]) (4.47.2)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dis
t-packages (from matplotlib->umap-learn[plot]) (1.4.5)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist
-packages (from matplotlib->umap-learn[plot]) (3.1.1)
Requirement already satisfied: networkx>=2.2 in /usr/local/lib/python3.10/dist-pa
ckages (from scikit-image->umap-learn[plot]) (3.2.1)
Requirement already satisfied: imageio>=2.4.1 in /usr/local/lib/python3.10/dist-p
ackages (from scikit-image->umap-learn[plot]) (2.31.6)
Requirement already satisfied: tifffile>=2019.7.26 in /usr/local/lib/python3.10/d
ist-packages (from scikit-image->umap-learn[plot]) (2023.12.9)
Requirement already satisfied: PyWavelets>=1.1.1 in /usr/local/lib/python3.10/dis
t-packages (from scikit-image->umap-learn[plot]) (1.5.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-
packages (from Jinja2>=2.9->bokeh->umap-learn[plot]) (2.1.4)
```

```
Requirement already satisfied: markdown in /usr/local/lib/python3.10/dist-package
s (from panel>=0.13.1->holoviews->umap-learn[plot]) (3.5.2)
Requirement already satisfied: markdown-it-py in /usr/local/lib/python3.10/dist-p
ackages (from panel>=0.13.1->holoviews->umap-learn[plot]) (3.0.0)
Requirement already satisfied: linkify-it-py in /usr/local/lib/python3.10/dist-pa
ckages (from panel>=0.13.1->holoviews->umap-learn[plot]) (2.0.2)
Requirement already satisfied: mdit-py-plugins in /usr/local/lib/python3.10/dist-
packages (from panel>=0.13.1->holoviews->umap-learn[plot]) (0.4.0)
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages
(from panel>=0.13.1->holoviews->umap-learn[plot]) (6.1.0)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.10/dis
t-packages (from panel>=0.13.1->holoviews->umap-learn[plot]) (4.5.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-package
s (from python-dateutil>=2.8.1->pandas->umap-learn[plot]) (1.16.0)
Requirement already satisfied: click>=8.0 in /usr/local/lib/python3.10/dist-packa
ges (from dask->datashader->umap-learn[plot]) (8.1.7)
Requirement already satisfied: cloudpickle>=1.5.0 in /usr/local/lib/python3.10/di
st-packages (from dask->datashader->umap-learn[plot]) (2.2.1)
Requirement already satisfied: fsspec>=2021.09.0 in /usr/local/lib/python3.10/dis
t-packages (from dask->datashader->umap-learn[plot]) (2023.6.0)
Requirement already satisfied: partd>=1.2.0 in /usr/local/lib/python3.10/dist-pac
kages (from dask->datashader->umap-learn[plot]) (1.4.1)
Requirement already satisfied: importlib-metadata>=4.13.0 in /usr/local/lib/pytho
n3.10/dist-packages (from dask->datashader->umap-learn[plot]) (7.0.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python
3.10/dist-packages (from requests->datashader->umap-learn[plot]) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-pac
kages (from requests->datashader->umap-learn[plot]) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/di
st-packages (from requests->datashader->umap-learn[plot]) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/di
st-packages (from requests->datashader->umap-learn[plot]) (2023.11.17)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.10/dist-packag
es (from importlib-metadata>=4.13.0->dask->datashader->umap-learn[plot]) (3.17.0)
Requirement already satisfied: locket in /usr/local/lib/python3.10/dist-packages
(from partd>=1.2.0->dask->datashader->umap-learn[plot]) (1.0.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-pac
kages (from bleach->panel>=0.13.1->holoviews->umap-learn[plot]) (0.5.1)
Requirement already satisfied: uc-micro-py in /usr/local/lib/python3.10/dist-pack
ages (from linkify-it-py->panel>=0.13.1->holoviews->umap-learn[plot]) (1.0.2)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-packa
ges (from markdown-it-py->panel>=0.13.1->holoviews->umap-learn[plot]) (0.1.2)
Requirement already satisfied: holoviews in /usr/local/lib/python3.10/dist-packag
es (1.17.1)
Requirement already satisfied: param<3.0,>=1.12.0 in /usr/local/lib/python3.10/di
st-packages (from holoviews) (2.0.2)
Requirement already satisfied: numpy>=1.0 in /usr/local/lib/python3.10/dist-packa
ges (from holoviews) (1.23.5)
Requirement already satisfied: pyviz-comms>=0.7.4 in /usr/local/lib/python3.10/di
st-packages (from holoviews) (3.0.1)
Requirement already satisfied: panel>=0.13.1 in /usr/local/lib/python3.10/dist-pa
ckages (from holoviews) (1.3.8)
Requirement already satisfied: colorcet in /usr/local/lib/python3.10/dist-package
s (from holoviews) (3.0.1)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packag
es (from holoviews) (23.2)
Requirement already satisfied: pandas>=0.20.0 in /usr/local/lib/python3.10/dist-p
ackages (from holoviews) (1.5.3)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.1
0/dist-packages (from pandas>=0.20.0->holoviews) (2.8.2)
```

```
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-pac
kages (from pandas>=0.20.0->holoviews) (2023.3.post1)
Requirement already satisfied: bokeh<3.4.0,>=3.2.0 in /usr/local/lib/python3.10/d
ist-packages (from panel>=0.13.1->holoviews) (3.3.4)
Requirement already satisfied: xyzservices>=2021.09.1 in /usr/local/lib/python3.1
0/dist-packages (from panel>=0.13.1->holoviews) (2023.10.1)
Requirement already satisfied: markdown in /usr/local/lib/python3.10/dist-package
s (from panel>=0.13.1->holoviews) (3.5.2)
Requirement already satisfied: markdown-it-py in /usr/local/lib/python3.10/dist-p
ackages (from panel>=0.13.1->holoviews) (3.0.0)
Requirement already satisfied: linkify-it-py in /usr/local/lib/python3.10/dist-pa
ckages (from panel>=0.13.1->holoviews) (2.0.2)
Requirement already satisfied: mdit-py-plugins in /usr/local/lib/python3.10/dist-
packages (from panel>=0.13.1->holoviews) (0.4.0)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-package
s (from panel>=0.13.1->holoviews) (2.31.0)
Requirement already satisfied: tqdm>=4.48.0 in /usr/local/lib/python3.10/dist-pac
kages (from panel>=0.13.1->holoviews) (4.66.1)
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages
(from panel>=0.13.1->holoviews) (6.1.0)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.10/dis
t-packages (from panel>=0.13.1->holoviews) (4.5.0)
Requirement already satisfied: pyct>=0.4.4 in /usr/local/lib/python3.10/dist-pack
ages (from colorcet->holoviews) (0.5.0)
Requirement already satisfied: Jinja2>=2.9 in /usr/local/lib/python3.10/dist-pack
ages (from bokeh<3.4.0,>=3.2.0->panel>=0.13.1->holoviews) (3.1.3)
Requirement already satisfied: contourpy>=1 in /usr/local/lib/python3.10/dist-pac
kages (from bokeh<3.4.0,>=3.2.0->panel>=0.13.1->holoviews) (1.2.0)
Requirement already satisfied: pillow>=7.1.0 in /usr/local/lib/python3.10/dist-pa
ckages (from bokeh<3.4.0,>=3.2.0->panel>=0.13.1->holoviews) (9.4.0)
Requirement already satisfied: PyYAML>=3.10 in /usr/local/lib/python3.10/dist-pac
kages (from bokeh<3.4.0,>=3.2.0->panel>=0.13.1->holoviews) (6.0.1)
Requirement already satisfied: tornado>=5.1 in /usr/local/lib/python3.10/dist-pac
kages (from bokeh<3.4.0,>=3.2.0->panel>=0.13.1->holoviews) (6.3.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-package
s (from python-dateutil>=2.8.1->pandas>=0.20.0->holoviews) (1.16.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-pac
kages (from bleach->panel>=0.13.1->holoviews) (0.5.1)
Requirement already satisfied: uc-micro-py in /usr/local/lib/python3.10/dist-pack
ages (from linkify-it-py->panel>=0.13.1->holoviews) (1.0.2)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-packa
ges (from markdown-it-py->panel>=0.13.1->holoviews) (0.1.2)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python
3.10/dist-packages (from requests->panel>=0.13.1->holoviews) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-pac
kages (from requests->panel>=0.13.1->holoviews) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/di
st-packages (from requests->panel>=0.13.1->holoviews) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/di
st-packages (from requests->panel>=0.13.1->holoviews) (2023.11.17)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-
packages (from Jinja2>=2.9->bokeh<3.4.0,>=3.2.0->panel>=0.13.1->holoviews) (2.1.
4)
Requirement already satisfied: ipykernel in /usr/local/lib/python3.10/dist-packag
es (6.29.0)
Requirement already satisfied: comm>=0.1.1 in /usr/local/lib/python3.10/dist-pack
ages (from ipykernel) (0.2.1)
Requirement already satisfied: debugpy>=1.6.5 in /usr/local/lib/python3.10/dist-p
ackages (from ipykernel) (1.6.6)
Requirement already satisfied: ipython>=7.23.1 in /usr/local/lib/python3.10/dist-
```

```
packages (from ipykernel) (7.34.0)
Requirement already satisfied: jupyter-client>=6.1.12 in /usr/local/lib/python3.1
0/dist-packages (from ipykernel) (6.1.12)
Requirement already satisfied: jupyter-core!=5.0.*,>=4.12 in /usr/local/lib/pytho
n3.10/dist-packages (from ipykernel) (5.7.1)
Requirement already satisfied: matplotlib-inline>=0.1 in /usr/local/lib/python3.1
0/dist-packages (from ipykernel) (0.1.6)
Requirement already satisfied: nest-asyncio in /usr/local/lib/python3.10/dist-pac
kages (from ipykernel) (1.6.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packag
es (from ipykernel) (23.2)
Requirement already satisfied: psutil in /usr/local/lib/python3.10/dist-packages
(from ipykernel) (5.9.5)
Requirement already satisfied: pyzmq>=24 in /usr/local/lib/python3.10/dist-packag
es (from ipykernel) (25.1.2)
Requirement already satisfied: tornado>=6.1 in /usr/local/lib/python3.10/dist-pac
kages (from ipykernel) (6.3.2)
Requirement already satisfied: traitlets>=5.4.0 in /usr/local/lib/python3.10/dist
-packages (from ipykernel) (5.7.1)
Requirement already satisfied: setuptools>=18.5 in /usr/local/lib/python3.10/dist
-packages (from ipython>=7.23.1->ipykernel) (67.7.2)
Requirement already satisfied: jedi>=0.16 in /usr/local/lib/python3.10/dist-packa
ges (from ipython>=7.23.1->ipykernel) (0.19.1)
Requirement already satisfied: decorator in /usr/local/lib/python3.10/dist-packag
es (from ipython>=7.23.1->ipykernel) (4.4.2)
Requirement already satisfied: pickleshare in /usr/local/lib/python3.10/dist-pack
ages (from ipython>=7.23.1->ipykernel) (0.7.5)
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in /u
sr/local/lib/python3.10/dist-packages (from ipython>=7.23.1->ipykernel) (3.0.43)
Requirement already satisfied: pygments in /usr/local/lib/python3.10/dist-package
s (from ipython>=7.23.1->ipykernel) (2.16.1)
Requirement already satisfied: backcall in /usr/local/lib/python3.10/dist-package
s (from ipython>=7.23.1->ipykernel) (0.2.0)
Requirement already satisfied: pexpect>4.3 in /usr/local/lib/python3.10/dist-pack
ages (from ipython>=7.23.1->ipykernel) (4.9.0)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.10/
dist-packages (from jupyter-client>=6.1.12->ipykernel) (2.8.2)
Requirement already satisfied: platformdirs>=2.5 in /usr/local/lib/python3.10/dis
t-packages (from jupyter-core!=5.0.*,>=4.12->ipykernel) (4.1.0)
Requirement already satisfied: parso<0.9.0,>=0.8.3 in /usr/local/lib/python3.10/d
ist-packages (from jedi>=0.16->ipython>=7.23.1->ipykernel) (0.8.3)
Requirement already satisfied: ptyprocess>=0.5 in /usr/local/lib/python3.10/dist-
packages (from pexpect>4.3->ipython>=7.23.1->ipykernel) (0.7.0)
Requirement already satisfied: wcwidth in /usr/local/lib/python3.10/dist-packages
(from prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0->ipython>=7.23.1->ipykernel)
(0.2.13)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-package
s (from python-dateutil>=2.1->jupyter-client>=6.1.12->ipykernel) (1.16.0)
ERROR: Could not find a version that satisfies the requirement ClusterEnsembles
(from versions: none)
ERROR: No matching distribution found for ClusterEnsembles
```

In [ ]:

```python
import numpy as np
import sklearn
import nltk, string
import matplotlib.pyplot as plt
```

In [ ]:

```python
import itertools
import numpy as np
import matplotlib.pyplot as plt
```

```python
import matplotlib.colors as colors
def plot_mat(mat, xticklabels = None, yticklabels = None, pic_fname = None, size
             colorbar = True, grid = 'k', xlabel = None, ylabel = None, title =
    if size == (-1, -1):
        size = (mat.shape[1] / 3, mat.shape[0] / 3)

    fig = plt.figure(figsize=size)
    ax = fig.add_subplot(1,1,1)

    # im = ax.imshow(mat, cmap=plt.cm.Blues)
    im = ax.pcolor(mat, cmap=plt.cm.Blues, linestyle='-', linewidth=0.5, edgecol

    if colorbar:
        plt.colorbar(im,fraction=0.046, pad=0.06)
    # tick_marks = np.arange(len(classes))
    # Ticks
    lda_num_topics = mat.shape[0]
    nmf_num_topics = mat.shape[1]
    yticks = np.arange(lda_num_topics)
    xticks = np.arange(nmf_num_topics)
    ax.set_xticks(xticks + 0.5)
    ax.set_yticks(yticks + 0.5)
    if xticklabels is None:
        xticklabels = [str(i) for i in xticks]
    if yticklabels is None:
        yticklabels = [str(i) for i in yticks]
    ax.set_xticklabels(xticklabels)
    ax.set_yticklabels(yticklabels)

    # Minor ticks
    # ax.set_xticks(xticks, minor=True);
    # ax.set_yticks(yticks, minor=True);
    # ax.set_xticklabels([], minor=True)
    # ax.set_yticklabels([], minor=True)

    # ax.grid(which='minor', color='k', linestyle='-', linewidth=0.5)

    # tick labels on all four sides
    ax.tick_params(labelright = True, labeltop = False)

    if ylabel:
        plt.ylabel(ylabel, fontsize=15)
    if xlabel:
        plt.xlabel(xlabel, fontsize=15)
    if title:
        plt.title(title, fontsize=15)

    # im = ax.imshow(mat, interpolation='nearest', cmap=plt.cm.Blues)
    ax.invert_yaxis()

    # thresh = mat.max() / 2

    def show_values(pc, fmt="%.3f", **kw):
        pc.update_scalarmappable()
        ax = pc.axes
        for p, color, value in itertools.zip_longest(pc.get_paths(), pc.get_face
            x, y = p.vertices[:-2, :].mean(0)
            if np.all(color[:3] > 0.5):
                color = (0.0, 0.0, 0.0)
            else:
```

```
                color = (1.0, 1.0, 1.0)
            ax.text(x, y, fmt % value, ha="center", va="center", color=color, **

    if if_show_values:
        show_values(im)
    # for i, j in itertools.product(range(mat.shape[0]), range(mat.shape[1])):
    #     ax.text(j, i, "{:.2f}".format(mat[i, j]), fontsize = 4,
    #             horizontalalignment="center",
    #             color="white" if mat[i, j] > thresh else "black")

    plt.tight_layout()
    if pic_fname:
        plt.savefig(pic_fname, dpi=300, transparent=True)
    plt.show()
    plt.close()
```

In [ ]:
```
from sklearn.datasets import fetch_20newsgroups

# get dataset
categories = ['comp.graphics', 'comp.os.ms-windows.misc', 'comp.sys.ibm.pc.hardw
              'rec.autos', 'rec.motorcycles','rec.sport.baseball', 'rec.spor
newsgroups = fetch_20newsgroups(subset = 'train',categories=categories, remove=(
```

In [ ]:
```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer

# count vectorizer on corpus
tf_vectorizer = CountVectorizer(min_df = 3, stop_words='english')
newsgroups_vectorized = tf_vectorizer.fit_transform(newsgroups.data)

# count vector to TF-IDF
transformer = TfidfTransformer()
newsgroups_tfidf = transformer.fit_transform(newsgroups_vectorized)

print('TF-IDF Dimensions: ', newsgroups_tfidf.shape)
```

```
TF-IDF Dimensions:  (4732, 17131)
```

Question 1

Dimensions of the TF-IDF matrix is (4732, 17131)

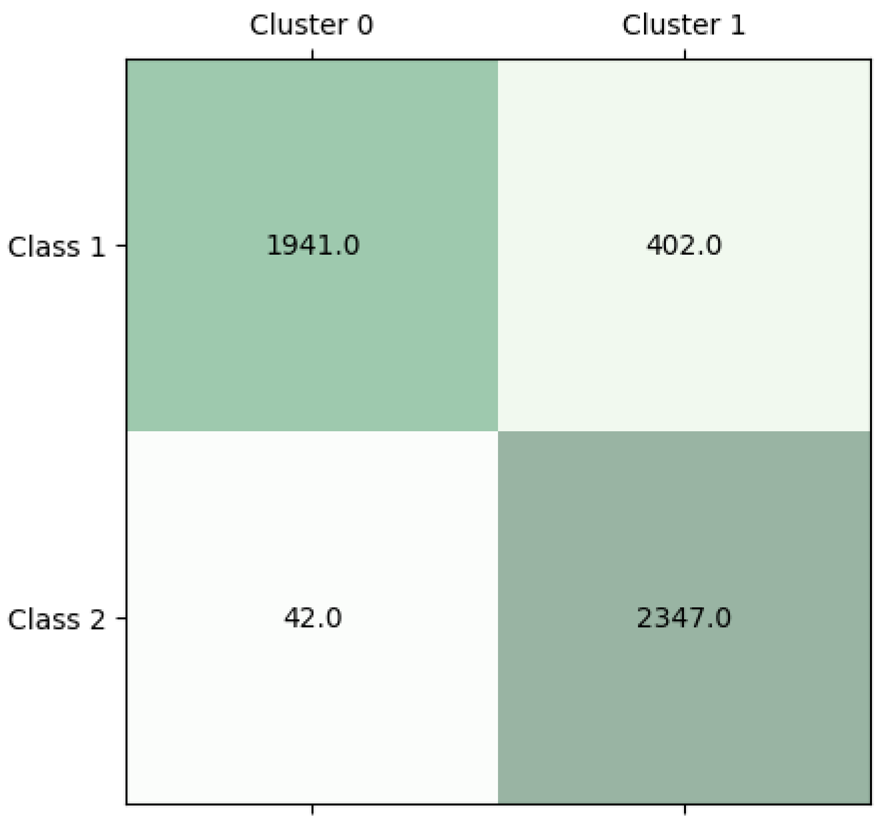
In [ ]:
```
from sklearn.cluster import KMeans
from sklearn.metrics.cluster import contingency_matrix
import matplotlib.pyplot as plt
import numpy as np

# Get clusters
kmeans = KMeans(random_state=0, n_clusters=2, max_iter=1000, n_init=30).fit(news
```

In [ ]:
```
label_kmeans = []
for label in newsgroups.target:
    if label in [0, 1, 2, 3]:
        label_kmeans.append(0)
    else:
        label_kmeans.append(1)
contingency_table = contingency_matrix(label_kmeans, kmeans.labels_)
print('Contingency Table:  ', '\n', contingency_table)
```

```
Contingency Table:
 [[1941  402]
 [  42 2347]]
```

Question 2

In [ ]:
```
from sklearn import metrics

# plot contingency matrix
plt.matshow(contingency_table, cmap=plt.cm.Greens, alpha=0.4)
for (i, j), z in np.ndenumerate(contingency_table):
    plt.text(j, i, '{:0.1f}'.format(z), ha='center', va='center')
plt.xticks(range(2), ['Cluster 0', 'Cluster 1'])
plt.yticks(range(2), ['Class 1', 'Class 2'])
plt.show()
```



*** Q2 Answer: ***

The presented contingency table illustrates the outcomes of clustering. From the prominent diagonal pattern, we deduce an association between group 1 and category 2, as well as group 0 and category 1. Considering that we configured Kmeans with 2 clusters, aligning with the 2 categories in our data, the contingency matrix is expected to be square. Any discrepancy between the cluster count set in Kmeans and the data's category count would result in a non-square contingency matrix.

Question 3

In [ ]:
```
from sklearn.metrics import cluster

print("Homogeneity score: %0.3f" % cluster.homogeneity_score(label_kmeans, kmean
```

```
print("Completeness score: %0.3f" % cluster.completeness_score(label_kmeans, kme
print("V-measure score: %0.3f" % cluster.v_measure_score(label_kmeans, kmeans.la
print("Adjusted Rand Index: %0.3f" % cluster.adjusted_rand_score(label_kmeans, k
print("Adjusted mutual information score: %0.3f" % cluster.adjusted_mutual_info_
```

```
Homogeneity score: 0.589
Completeness score: 0.601
V-measure score: 0.595
Adjusted Rand Index: 0.660
Adjusted mutual information score: 0.595
```

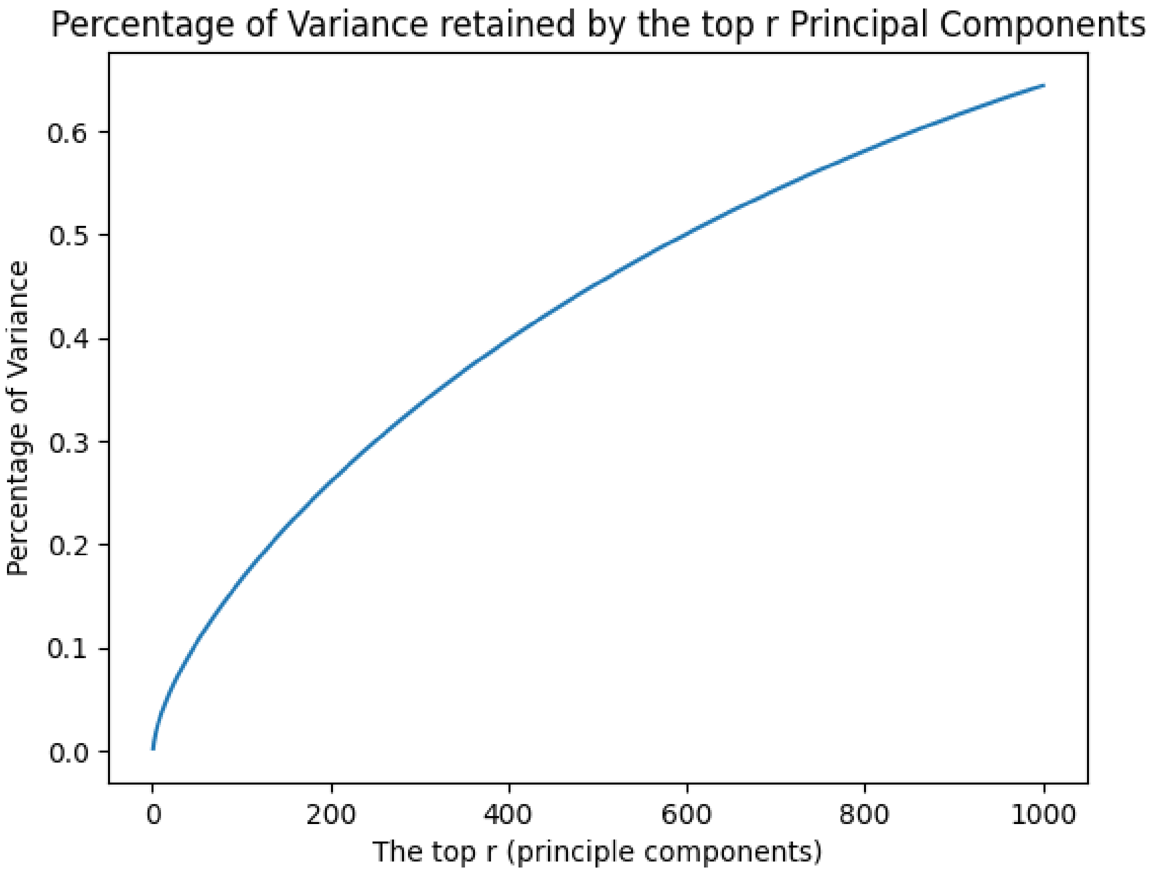
Clustering with Dense Text Representations

1. Generate dense representations for better K-Means Clustering

Question 4

In [ ]:

```
from sklearn.decomposition import TruncatedSVD

# get principle components
svd = TruncatedSVD(n_components=1000, random_state=0)
newsgroups_lsi = svd.fit_transform(newsgroups_tfidf)

# get explained variance ratio
x = np.linspace(1, 1000, 1000)
ratio = svd.explained_variance_ratio_.cumsum()

# plot explained variance ratio
plt.plot(x, ratio)
plt.ylabel('Percentage of Variance')
plt.xlabel('The top r (principle components)')
plt.title('Percentage of Variance retained by the top r Principal Components')
plt.show()
```

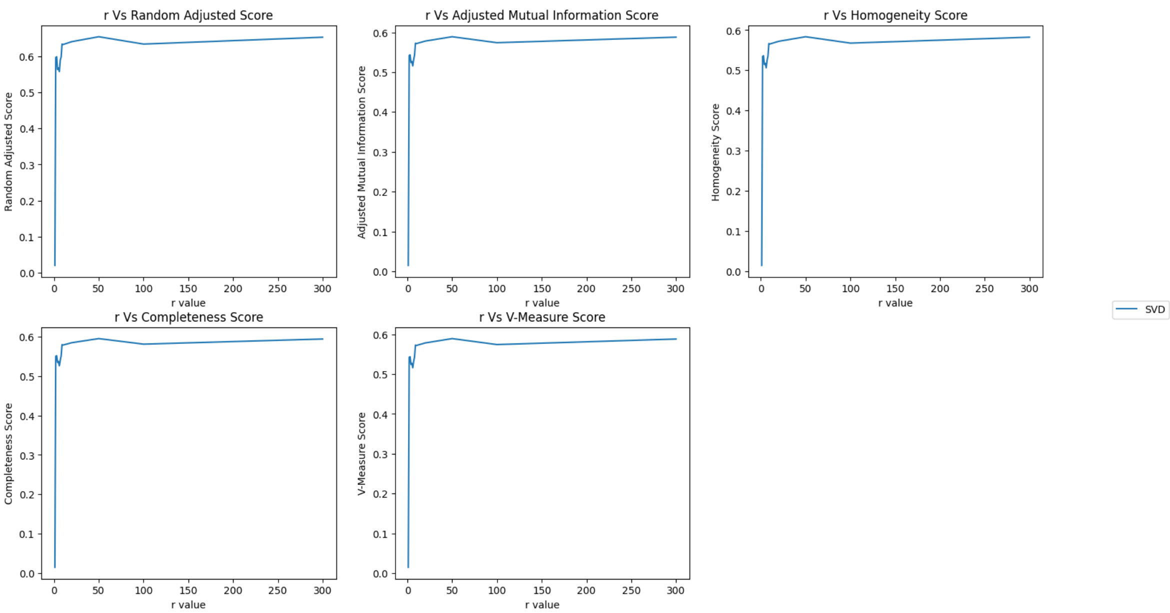


Question 5

In [ ]:
```
from sklearn.metrics import adjusted_rand_score, adjusted_mutual_info_score, hom
import statistics

def calculate_svd_scores(r, k, X, y):
    svd_adj_rand_score = []
    svd_adj_mutual_score = []
    svd_hom_score = []
    svd_comp_score = []
    svd_v_score = []

    for dim in r:
        svd = TruncatedSVD(n_components=dim, random_state=0)
        truncated_svd = svd.fit_transform(X)
        kmeans = KMeans(random_state=0, n_clusters=k, max_iter=1000, n_init=30)
        kmeans.fit(truncated_svd)

        svd_adj_rand_score.append(adjusted_rand_score(y, kmeans.labels_))
        svd_adj_mutual_score.append(adjusted_mutual_info_score(y, kmeans.labels_
        svd_hom_score.append(homogeneity_score(y, kmeans.labels_))
        svd_comp_score.append(completeness_score(y, kmeans.labels_))
        svd_v_score.append(v_measure_score(y, kmeans.labels_))

    return svd_adj_rand_score, svd_adj_mutual_score, svd_hom_score, svd_comp_sco

def plot_metrics_vs_r(r, svd_adj_rand_score, svd_adj_mutual_score, svd_hom_score
    fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(18, 10))

    def plot(ax, data, label, title, xlabel, ylabel):
        ax.plot(r, data, label=label)
        ax.set_title(title)
```

```
        ax.set_xlabel(xlabel)
        ax.set_ylabel(ylabel)

    plot(axes[0, 0], svd_adj_rand_score, 'SVD', 'r Vs Random Adjusted Score', 'r
    plot(axes[0, 1], svd_adj_mutual_score, 'SVD', 'r Vs Adjusted Mutual Informat
    plot(axes[0, 2], svd_hom_score, 'SVD', 'r Vs Homogeneity Score', 'r value',
    plot(axes[1, 0], svd_comp_score, 'SVD', 'r Vs Completeness Score', 'r value'
    plot(axes[1, 1], svd_v_score, 'SVD', 'r Vs V-Measure Score', 'r value', 'V-M

    axes[1, 2].axis('off')
    fig.legend(['SVD'], loc='center right')

    plt.show()

def find_best_r_value(scores):
    argmaxes = [i.index(max(i)) for i in scores]
    best_r_ind = round(statistics.mode(argmaxes))
    return best_r_ind
```

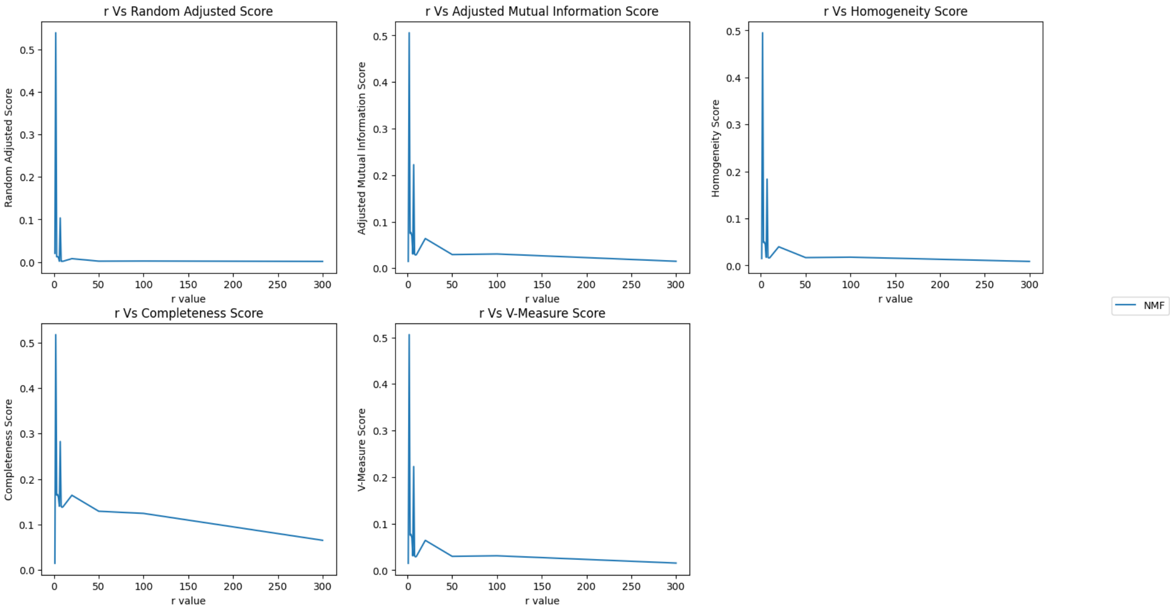
In [ ]:
```
r = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 20, 50, 100, 300]
k = 2

svd_scores = calculate_svd_scores(r, k,  newsgroups_tfidf, label_kmeans)

svd_adj_rand_score, svd_adj_mutual_score, svd_hom_score, svd_comp_score, svd_v_s

plot_metrics_vs_r(r, svd_adj_rand_score, svd_adj_mutual_score, svd_hom_score, sv

best_svd_r_value = r[find_best_r_value(svd_scores)]
print('Best SVD r value:', best_svd_r_value)
```




```
Best SVD r value: 50
```

In [ ]:
```
from sklearn.decomposition import NMF
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn import metrics
import statistics

def calculate_nmf_scores(r_values, k, X, y):
    adj_rand_score = []
    adj_mutual_score = []
```

```
    hom_score = []
    comp_score = []
    v_score = []

    for dim in r_values:
        nmf = NMF(n_components=dim, init='random', random_state=0, max_iter=500)
        trunc_nmf = nmf.fit_transform(X)
        kmeans = KMeans(random_state=0, n_clusters=k, max_iter=1000, n_init=30)
        kmeans.fit(trunc_nmf)
        adj_rand_score.append(metrics.adjusted_rand_score(y, kmeans.labels_))
        adj_mutual_score.append(metrics.adjusted_mutual_info_score(y, kmeans.lab
        hom_score.append(metrics.homogeneity_score(y, kmeans.labels_))
        comp_score.append(metrics.completeness_score(y, kmeans.labels_))
        v_score.append(metrics.v_measure_score(y, kmeans.labels_))

    return adj_rand_score, adj_mutual_score, hom_score, comp_score, v_score

def plot_nmf_scores(r_values, scores):
    fig, axes = plt.subplots(nrows=2, ncols=3, figsize=(18, 10))
    metrics_names = ['Random Adjusted', 'Adjusted Mutual Information', 'Homogene

    for i, metric in enumerate(scores):
        row, col = divmod(i, 3)
        axes[row, col].plot(r_values, metric, label='NMF')
        axes[row, col].set_title(f'r Vs {metrics_names[i]} Score')
        axes[row, col].set_xlabel('r value')
        axes[row, col].set_ylabel(f'{metrics_names[i]} Score')

    axes[1, 2].axis('off')
    fig.legend(['NMF'], loc='center right')
    plt.show()

def find_best_r_value(scores):
    argmaxes = [metric.index(max(metric)) for metric in scores]
    best_r_ind = round(statistics.mode(argmaxes))
    return best_r_ind

# Assuming you have r, newsgroups_tfidf, label_kmeans defined
r_values = r
k_clusters = 2

nmf_scores = calculate_nmf_scores(r_values, k_clusters, newsgroups_tfidf, label_
nmf_adj_rand_score, nmf_mutual_score, nmf_hom_score, nmf_comp_score, nmf_v_score

plot_nmf_scores(r_values, [nmf_adj_rand_score, nmf_mutual_score, nmf_hom_score,

nmf_score = [nmf_adj_rand_score, nmf_mutual_score, nmf_hom_score, nmf_comp_score
best_nmf_r_value = find_best_r_value(nmf_score)

print('Best NMF r value:', r[best_nmf_r_value])
```




```
Best NMF r value: 2
```

Q5 Answer:

A good choice of r for SVD is 50. A good choice of r for NMF is 2.

Question 6

While dimensionality reduction helps to deal with noisy data and shorten the running time of the algorithm, it may also lead to loss of information, including noise. Thus, as the dimensionality reduction parameter r increases, the accuracy of KMeans clustering may decrease because we truncate too much data. However, as r increases, we may observe that the clustering score initially increases and then decreases. The initial increase indicates that we have struck a good balance between scores and truncated data. An eventual decrease may indicate that as the truncated data increases, more noise is introduced, leading to inaccurate KMeans clustering and lower scores. Thus, we can observe a non-monotonic behavior of the measurements as r increases.

Question 7

In [ ]:
```
def print_average_metrics(method, hom_score, comp_score, v_score, adj_rand_score
    print(f"\n{method} Metrics:")
    print("Homogeneity: ", np.mean(hom_score))
    print("Completeness: ", np.mean(comp_score))
    print("V-measure: ", np.mean(v_score))
    print("Adjusted Rand-Index: ", np.mean(adj_rand_score))
    print("Adjusted Mutual Information: ", np.mean(adj_mutual_score))

print_average_metrics("SVD", svd_hom_score, svd_comp_score, svd_v_score, svd_adj
print_average_metrics("NMF", nmf_hom_score, nmf_comp_score, nmf_v_score, nmf_adj
```

```
SVD Metrics:
Homogeneity:  0.5086295760481007
Completeness:  0.5226573391383235
V-measure:  0.515542491271773
Adjusted Rand-Index:  0.567182526171996
Adjusted Mutual Information:  0.5154676533294708

NMF Metrics:
Homogeneity:  0.0704770686887141
Completeness:  0.16756615651774417
V-measure:  0.08748731407893089
Adjusted Rand-Index:  0.051330192251747724
Adjusted Mutual Information:  0.08726492696038733
```

Q7 Answer

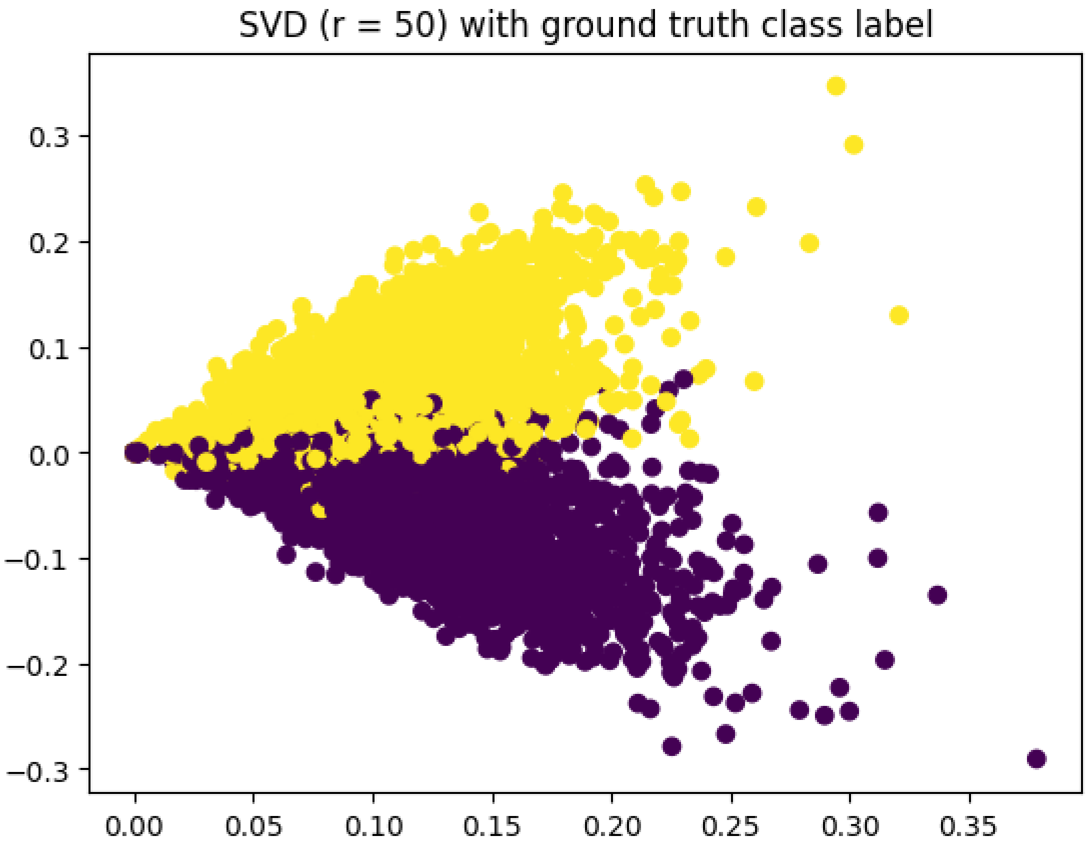
Both SVD and NMF metrics, on average, are worse than those computed in Question 3. However, SVD performs relatively better and is closer to the metrics from Question 3 compared to NMF.

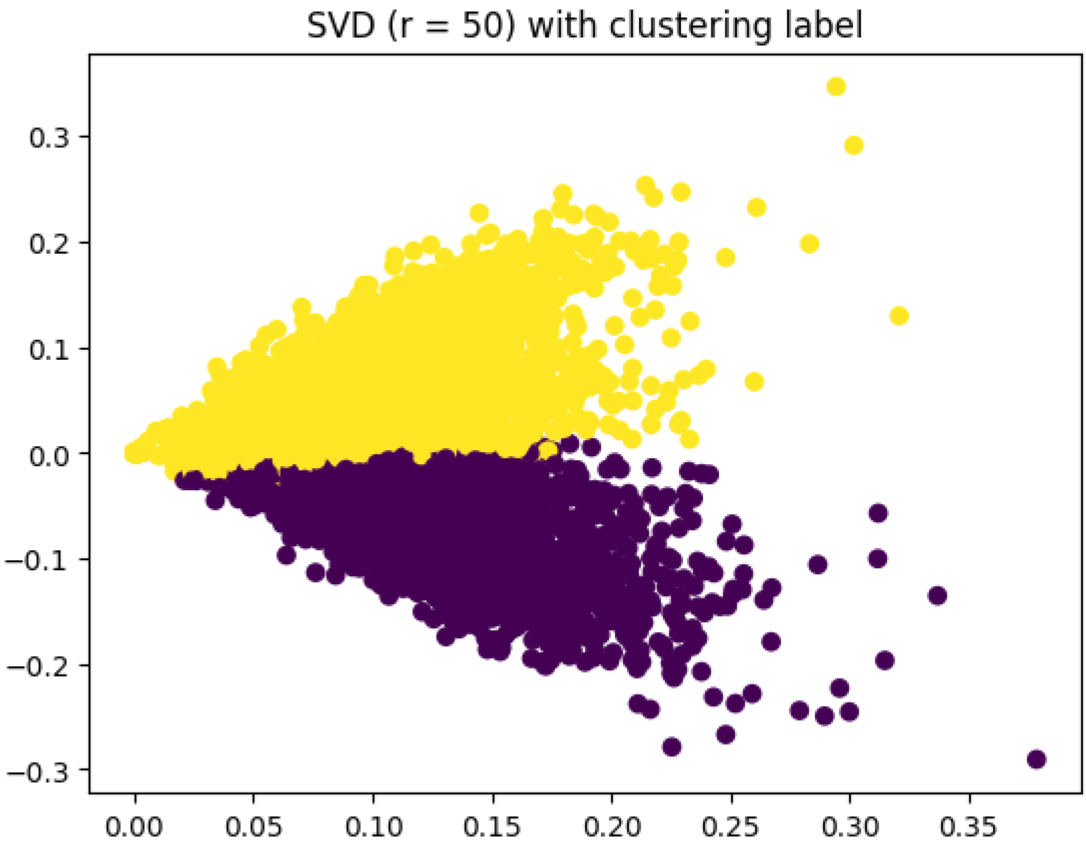
2. Visualize the clusters

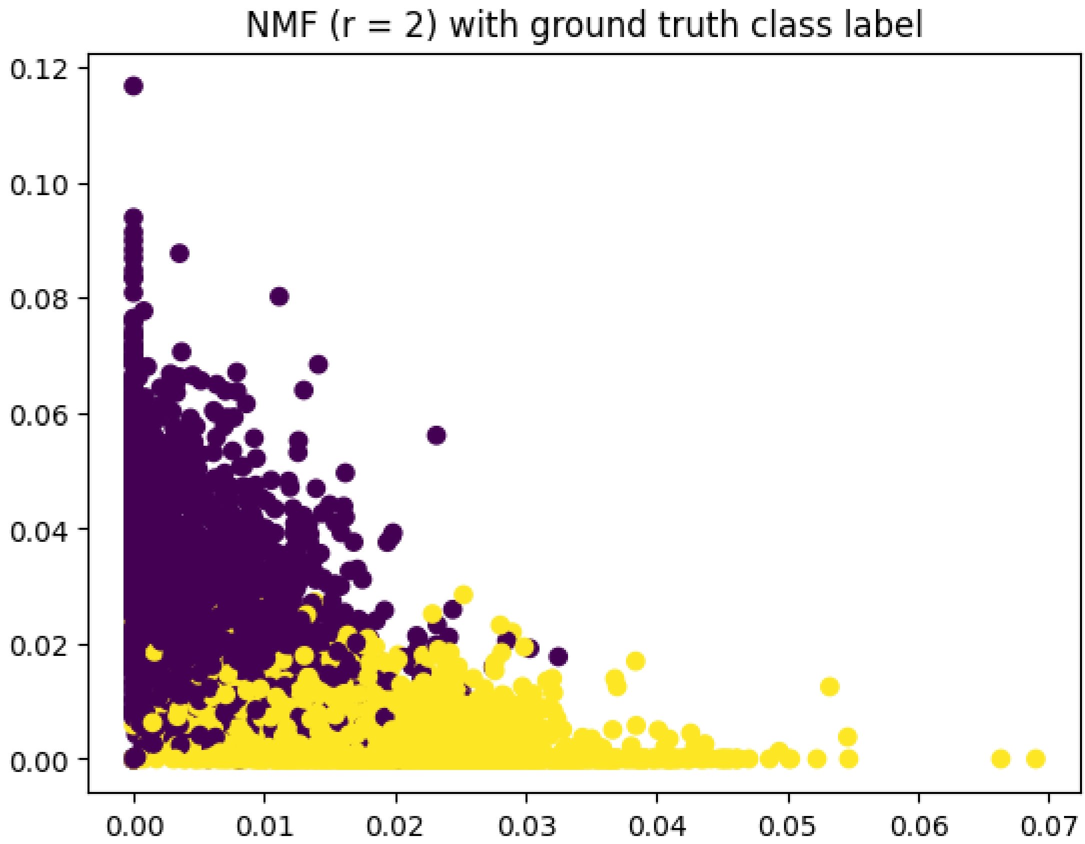
Question 8

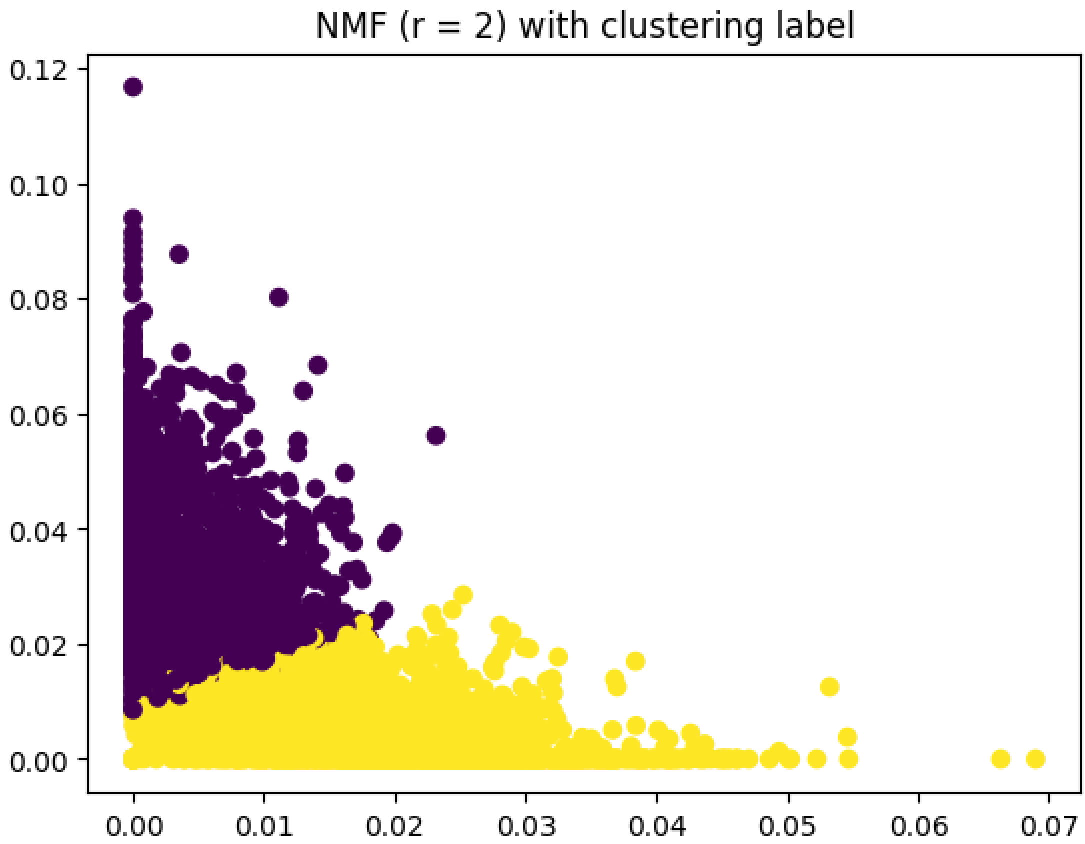
In [ ]:
```
from sklearn.decomposition import TruncatedSVD, NMF
import matplotlib.pyplot as plt

def perform_svd(data, components=50, random_state=42):
    svd_model = TruncatedSVD(n_components=components, random_state=random_state)
    svd_transformed = svd_model.fit_transform(data)
    return svd_transformed

def perform_nmf(data, components=2, random_state=0):
    nmf_model = NMF(n_components=components, init='random', random_state=random_
    nmf_transformed = nmf_model.fit_transform(data)
    return nmf_transformed

def plot_scatter(transformed_data, labels, title):
    plt.scatter(transformed_data[:, 0], transformed_data[:, 1], c=labels)
    plt.title(title)
    plt.show()
```

In [ ]:
```
svd_transformed_data = perform_svd(newsgroups_tfidf)
nmf_transformed_data = perform_nmf(newsgroups_tfidf)

plot_scatter(svd_transformed_data, label_kmeans, "SVD (r = 50) with ground truth
plot_scatter(svd_transformed_data, kmeans.labels_, "SVD (r = 50) with clustering
plot_scatter(nmf_transformed_data, label_kmeans, "NMF (r = 2) with ground truth
plot_scatter(nmf_transformed_data, kmeans.labels_, "NMF (r = 2) with clustering
```


SVD (r = 50) with ground truth class label
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35


SVD (r = 50) with clustering label
0.3
0.2
0.1
0.0
−0.1
−0.2
−0.3
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35





Question 9

The aforementioned graphs reveal a striking similarity between the clustered labels and the actual group labels. Nevertheless, the genuine group labels exhibit a greater level of overlap, a nuance not distinctly evident in the labeling graphs generated by NMF and SVD, where labeling boundaries are more clearly defined.

The data portrays a triangular distribution rather than a spherical one, with centroids of individual labels closely positioned. Simultaneously, outliers are present at a considerable distance from the primary clusters. Given that K-Means clustering assumes a spherical data distribution, this non-spherical data distribution poses a suboptimal scenario.

3. Clustering of the Entire 20 Classes

Question 10

In [ ]:
```
from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
import pandas as pd

# Load the dataset
news_dataset = fetch_20newsgroups(subset = 'all',shuffle = True, random_state =

def load_and_transform_dataset(min_df=3):

    # Create CountVectorizer and TfidfTransformer
    vectorizer = CountVectorizer(stop_words="english", min_df=min_df)
    transformer = TfidfTransformer(use_idf=True)

    # Transform the text data
    word_count_matrix = vectorizer.fit_transform(news_dataset.data)
    tfidf_matrix = transformer.fit_transform(word_count_matrix)
    tfidf_array = tfidf_matrix.toarray()

    # Get feature names from CountVectorizer
    feature_names = vectorizer.get_feature_names_out()

    # Create a DataFrame with the transformed data
    tfidf_dataframe = pd.DataFrame(data=tfidf_array, columns=feature_names)

    return tfidf_dataframe

# Example usage
tfidf_dataframe = load_and_transform_dataset()
print(tfidf_dataframe.shape)
```

```
(18846, 45365)
```

In [ ]:
```
from sklearn.decomposition import TruncatedSVD
from sklearn.cluster import KMeans
from sklearn.metrics import homogeneity_score, completeness_score, v_measure_scc

def calculate_best_svd_score(r_values, data, kmeans_clusters, target_labels):
    best_score_svd = 0
    best_r_svd = 0

    for r in r_values:
        print(r)
        svd_model = TruncatedSVD(n_components=r, random_state=42)
        svd_features = svd_model.fit_transform(data)
        kmeans_clusters.fit(svd_features)

        hs = homogeneity_score(target_labels, kmeans_clusters.labels_)
```

```
        cs = completeness_score(target_labels, kmeans_clusters.labels_)
        vms = v_measure_score(target_labels, kmeans_clusters.labels_)
        aris = adjusted_rand_score(target_labels, kmeans_clusters.labels_)
        amis = adjusted_mutual_info_score(target_labels, kmeans_clusters.labels_

        avg_svd_score = (hs + cs + vms + aris + amis) / 5

        print('Average Score: ' + str(avg_svd_score))

        if avg_svd_score > best_score_svd:
            best_score_svd = avg_svd_score
            best_r_svd = r

    return best_r_svd, best_score_svd

num_components = [1, 2, 3, 5, 10, 20, 50, 100, 300]
kmeans_cluster_model = KMeans(init='k-means++', max_iter=1000, n_clusters=20, n_

best_r_svd, best_svd_score = calculate_best_svd_score(num_components, tfidf_data

print('Best r in terms of average score: ' + str(best_r_svd))
print('Best SVD Score: ' + str(best_svd_score))
```

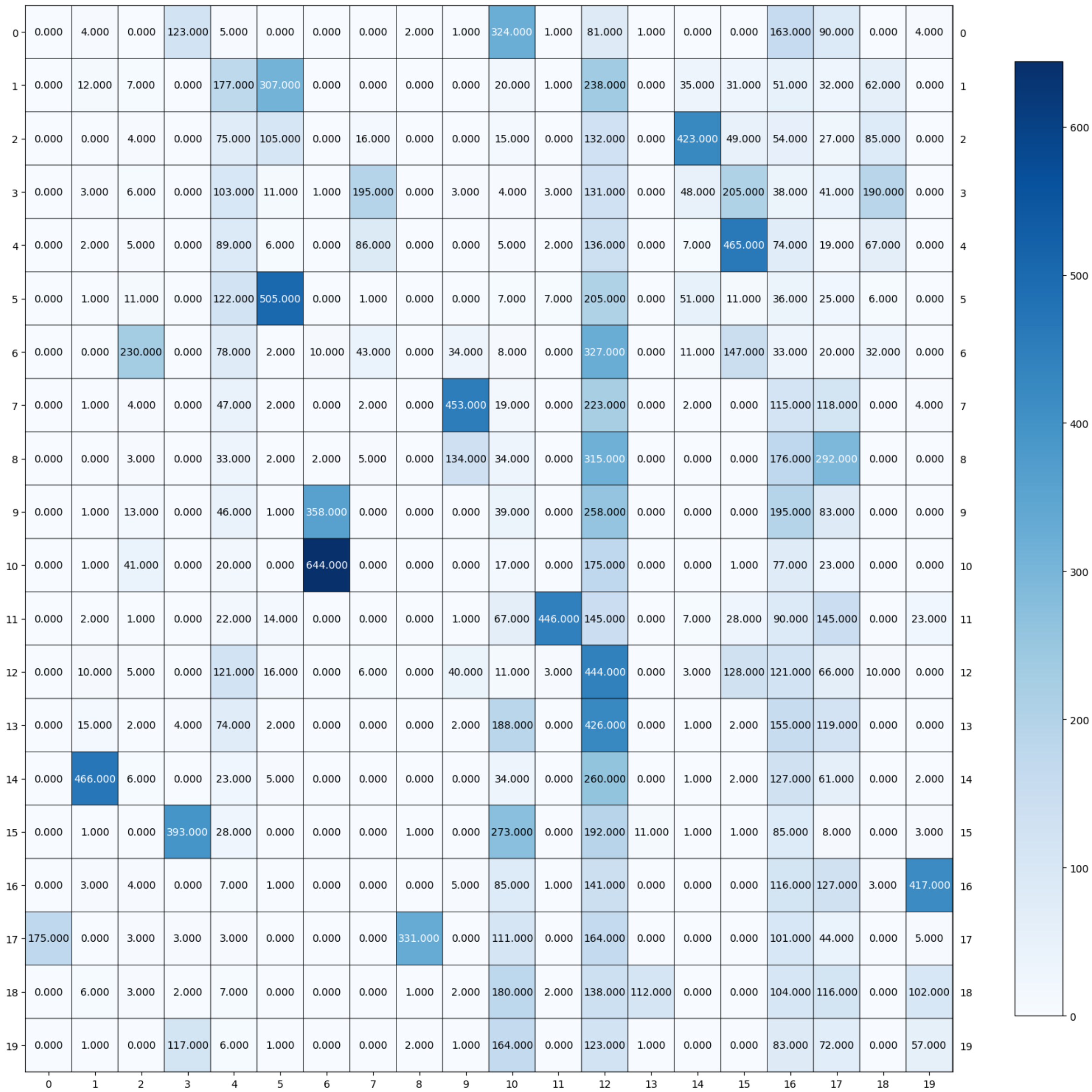
```
1
Average Score: 0.020699099772092347
2
Average Score: 0.1866356606182337
3
Average Score: 0.2210624664991326
5
Average Score: 0.2933199644577881
10
Average Score: 0.29342498645198073
20
Average Score: 0.30886229818530025
50
Average Score: 0.30086532474510075
100
Average Score: 0.30106677274936217
300
Average Score: 0.2783574580258681
Best r in terms of average score: 20
Best SVD Score: 0.30886229818530025
```

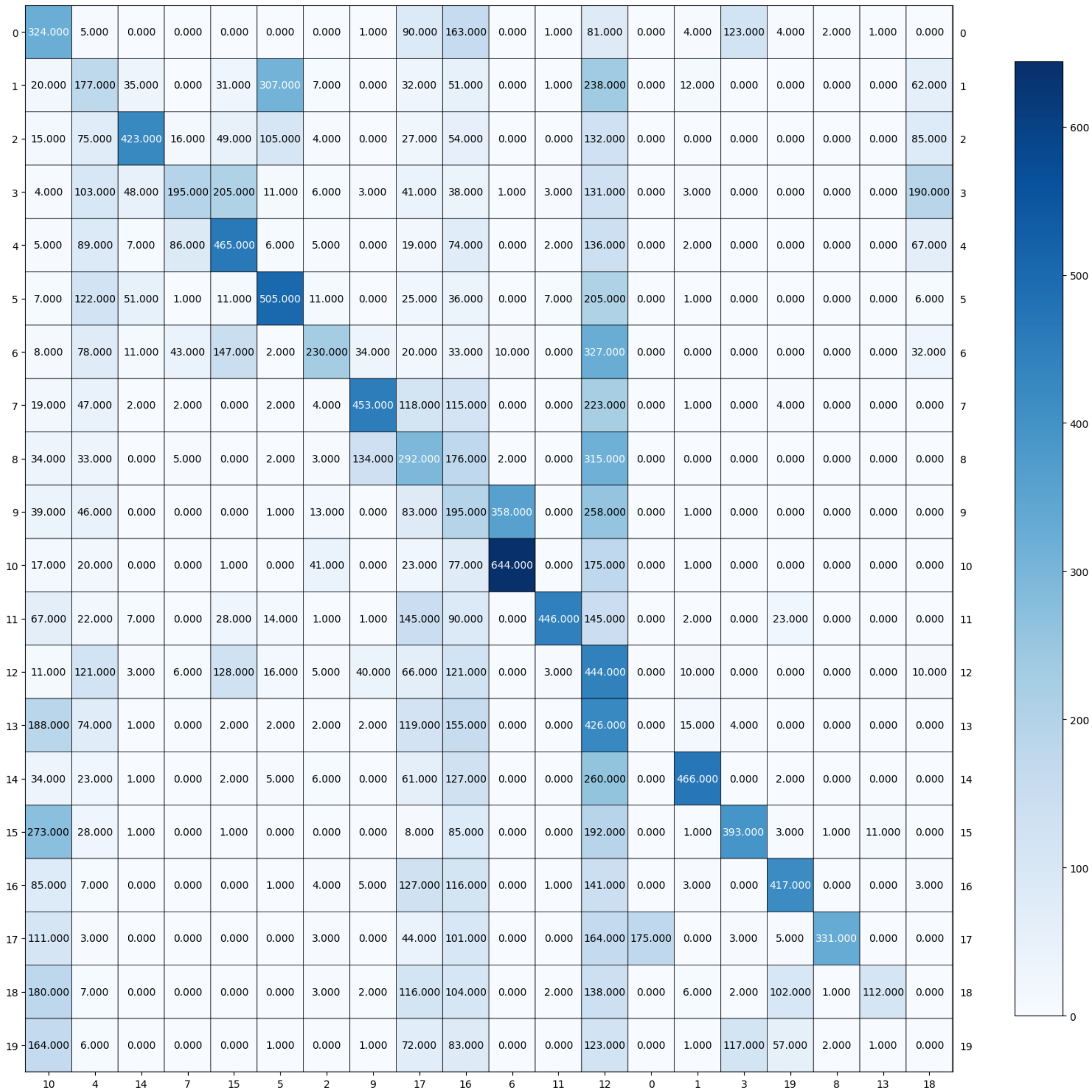
In [ ]:

```
from sklearn.decomposition import TruncatedSVD
from sklearn.cluster import KMeans
from sklearn.metrics import (
    homogeneity_score,
    completeness_score,
    v_measure_score,
    adjusted_rand_score,
    adjusted_mutual_info_score,
)
from sklearn.metrics import confusion_matrix
from scipy.optimize import linear_sum_assignment
from sklearn.metrics.cluster import contingency_matrix

def apply_svd_and_kmeans(data, num_components, kmeans_model):
    svd_transformer = TruncatedSVD(n_components=num_components, random_state=42)
    svd_features = svd_transformer.fit_transform(data)
```

```
    kmeans_model.fit(svd_features)

def evaluate_clustering_metrics(y_test, y_pred, name=""):
    print("Homogeneity score for %s: %f" % (name, homogeneity_score(y_test, y_pr
    print("Completeness score for %s: %f" % (name, completeness_score(y_test, y_
    print("V-measure score for %s: %f" % (name, v_measure_score(y_test, y_pred))
    print("Adjusted Rand Index score for %s: %f" % (name, adjusted_rand_score(y_
    print("Adjusted mutual information score for %s: %f" % (name, adjusted_mutua


def visualize_confusion_matrix(target_labels, predicted_labels):
    cm = confusion_matrix(target_labels, predicted_labels)
    rows, cols = linear_sum_assignment(cm, maximize=True)
    plot_mat(cm[rows[:, np.newaxis], cols], xticklabels=cols, yticklabels=rows,

# R = 20 - Average Score
svd_r = 20
svd_model = TruncatedSVD(n_components=svd_r, random_state=42)
words_count_svd = svd_model.fit_transform(tfidf_dataframe)

kmeans_model = KMeans(init='k-means++', max_iter=1000, n_clusters=20, n_init=30,
apply_svd_and_kmeans(tfidf_dataframe, svd_r, kmeans_model)

# Evaluate metrics
evaluate_clustering_metrics(news_dataset.target, kmeans_model.labels_, name="SVD

# Visualize confusion matrix
plot_mat(contingency_matrix(news_dataset.target, kmeans_model.labels_), size = (
visualize_confusion_matrix(news_dataset.target, kmeans_model.labels_)
```

```
Homogeneity score for SVD (r = 20): 0.336158
Completeness score for SVD (r = 20): 0.378021
V-measure score for SVD (r = 20): 0.355862
Adjusted Rand Index score for SVD (r = 20): 0.120619
Adjusted mutual information score for SVD (r = 20): 0.353652
```


0 0.000 4.000 0.000 123.000 5.000 0.000 0.000 0.000 2.000 1.000 324.000 1.000 81.000 1.000 0.000 0.000 163.000 90.000 0.000 4.000 0
1 0.000 12.000 7.000 0.000 177.000 307.000 0.000 0.000 0.000 0.000 20.000 1.000 238.000 0.000 35.000 31.000 51.000 32.000 62.000 0.000 1
2 0.000 0.000 4.000 0.000 75.000 105.000 0.000 16.000 0.000 0.000 15.000 0.000 132.000 0.000 423.000 49.000 54.000 27.000 85.000 0.000 2
3 0.000 3.000 6.000 0.000 103.000 11.000 1.000 195.000 0.000 3.000 4.000 3.000 131.000 0.000 48.000 205.000 38.000 41.000 190.000 0.000 3
4 0.000 2.000 5.000 0.000 89.000 6.000 0.000 86.000 0.000 0.000 5.000 2.000 136.000 0.000 7.000 465.000 74.000 19.000 67.000 0.000 4
5 0.000 1.000 11.000 0.000 122.000 505.000 0.000 1.000 0.000 0.000 7.000 7.000 205.000 0.000 51.000 11.000 36.000 25.000 6.000 0.000 5
6 0.000 0.000 230.000 0.000 78.000 2.000 10.000 43.000 0.000 34.000 8.000 0.000 327.000 0.000 11.000 147.000 33.000 20.000 32.000 0.000 6
7 0.000 1.000 4.000 0.000 47.000 2.000 0.000 2.000 0.000 453.000 19.000 0.000 223.000 0.000 2.000 0.000 115.000 118.000 0.000 4.000 7
8 0.000 0.000 3.000 0.000 33.000 2.000 2.000 5.000 0.000 134.000 34.000 0.000 315.000 0.000 0.000 0.000 176.000 292.000 0.000 0.000 8
9 0.000 1.000 13.000 0.000 46.000 1.000 358.000 0.000 0.000 0.000 39.000 0.000 258.000 0.000 0.000 0.000 195.000 83.000 0.000 0.000 9
10 0.000 1.000 41.000 0.000 20.000 0.000 644.000 0.000 0.000 0.000 17.000 0.000 175.000 0.000 0.000 1.000 77.000 23.000 0.000 0.000 10
11 0.000 2.000 1.000 0.000 22.000 14.000 0.000 0.000 0.000 1.000 67.000 446.000 145.000 0.000 7.000 28.000 90.000 145.000 0.000 23.000 11
12 0.000 10.000 5.000 0.000 121.000 16.000 0.000 6.000 0.000 40.000 11.000 3.000 444.000 0.000 3.000 128.000 121.000 66.000 10.000 0.000 12
13 0.000 15.000 2.000 4.000 74.000 2.000 0.000 0.000 0.000 2.000 188.000 0.000 426.000 0.000 1.000 2.000 155.000 119.000 0.000 0.000 13
14 0.000 466.000 6.000 0.000 23.000 5.000 0.000 0.000 0.000 0.000 34.000 0.000 260.000 0.000 1.000 2.000 127.000 61.000 0.000 2.000 14
15 0.000 1.000 0.000 393.000 28.000 0.000 0.000 0.000 1.000 0.000 273.000 0.000 192.000 11.000 1.000 1.000 85.000 8.000 0.000 3.000 15
16 0.000 3.000 4.000 0.000 7.000 1.000 0.000 0.000 0.000 5.000 85.000 1.000 141.000 0.000 0.000 0.000 116.000 127.000 3.000 417.000 16
17 175.000 0.000 3.000 3.000 3.000 0.000 0.000 0.000 331.000 0.000 111.000 0.000 164.000 0.000 0.000 0.000 101.000 44.000 0.000 5.000 17
18 0.000 6.000 3.000 2.000 7.000 0.000 0.000 0.000 1.000 2.000 180.000 2.000 138.000 112.000 0.000 0.000 104.000 116.000 0.000 102.000 18
19 0.000 1.000 0.000 117.000 6.000 1.000 0.000 0.000 2.000 1.000 164.000 0.000 123.000 1.000 0.000 0.000 83.000 72.000 0.000 57.000 19
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
600
500
400
300
200
100
0



In [ ]:

```
from sklearn.decomposition import NMF
from sklearn.cluster import KMeans
from sklearn.metrics import (
    homogeneity_score,
    completeness_score,
    v_measure_score,
    adjusted_rand_score,
    adjusted_mutual_info_score,
)
num_components = [1, 2, 3, 5, 10, 20, 50, 100, 300]
def calculate_best_nmf(components, data, target_labels, n_clusters=20, random_st
    best_nmf_score = 0
    best_nmf_r = 0

    kmeans_model = KMeans(init='k-means++', max_iter=100000, n_clusters=n_cluste

    for r in components:
        nmf_model = NMF(n_components=r, init='random', random_state=random_state
        words_count_nmf = nmf_model.fit_transform(data)
        kmeans_model.fit(words_count_nmf)

        hs = homogeneity_score(target_labels, kmeans_model.labels_)
        cs = completeness_score(target_labels, kmeans_model.labels_)
        vms = v_measure_score(target_labels, kmeans_model.labels_)
```

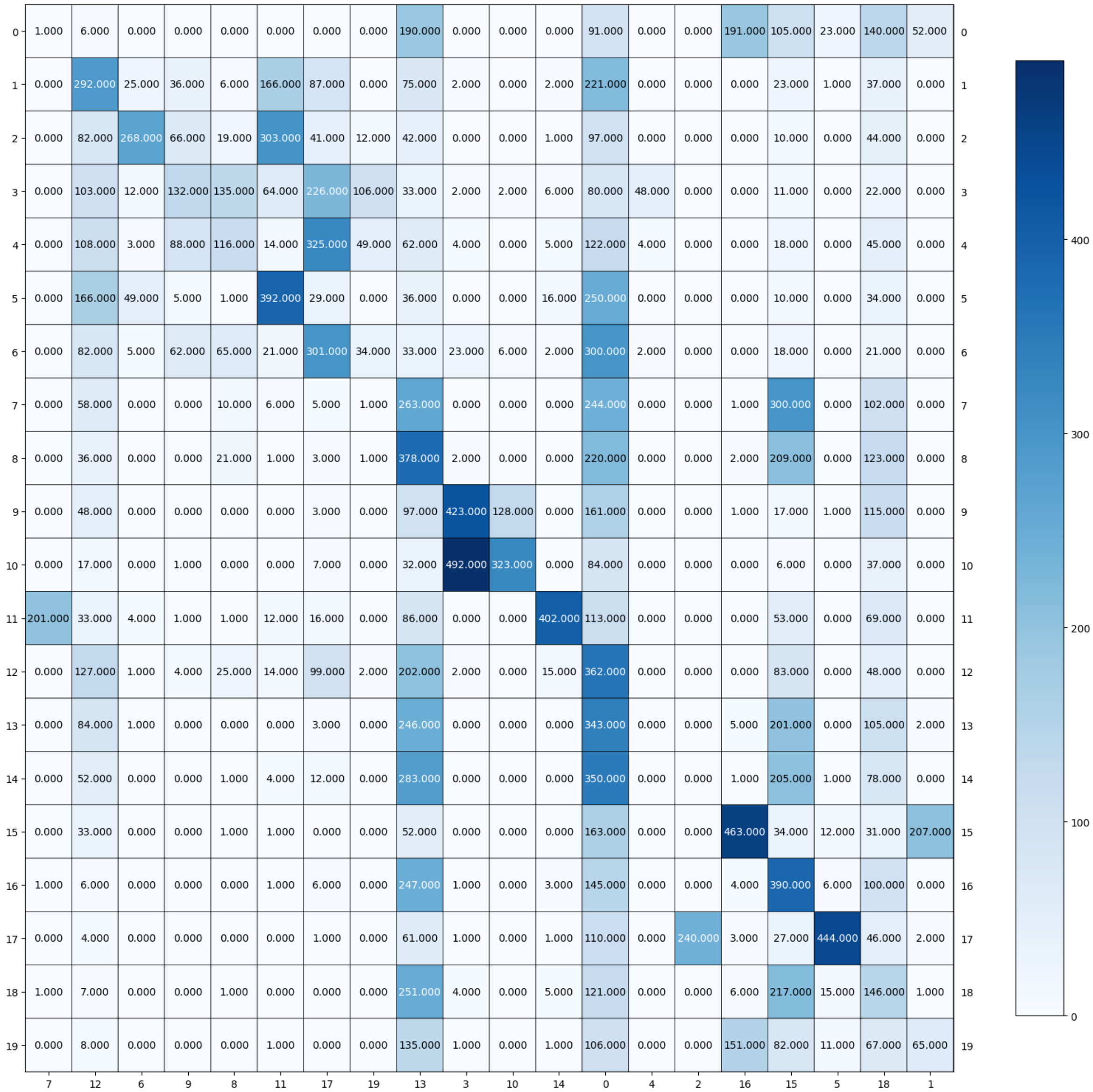
```
        aris = adjusted_rand_score(target_labels, kmeans_model.labels_)
        amis = adjusted_mutual_info_score(target_labels, kmeans_model.labels_)

        avg_score = (hs + cs + vms + aris + amis) / 5

        if avg_score > best_nmf_score:
            best_nmf_score = avg_score
            best_nmf_r = r

        print('Component ' + str(r) + ', ' + 'Average Score: ' + str(avg_score))

    return best_nmf_r, best_nmf_score

best_r_nmf, best_score_nmf = calculate_best_nmf(num_components, tfidf_dataframe,

print('Best r for NMF: ' + str(best_r_nmf))
print('Best NMF Score: ' + str(best_score_nmf))
```

```
Component 1, Average Score: 0.02076162215956203
Component 2, Average Score: 0.1699054207666734
Component 3, Average Score: 0.20177601465736045
Component 5, Average Score: 0.2396382853881612
Component 10, Average Score: 0.2650333039858369
Component 20, Average Score: 0.2627086703407877
Component 50, Average Score: 0.23521298076051372
Component 100, Average Score: 0.14074921062990936
Component 300, Average Score: 0.05024577283103636
Best r for NMF: 10
Best NMF Score: 0.2650333039858369
```

In [ ]:

```
from sklearn.metrics import confusion_matrix
from scipy.optimize import linear_sum_assignment
from sklearn.metrics import confusion_matrix
from scipy.optimize import linear_sum_assignment
from sklearn.decomposition import NMF
from sklearn.cluster import KMeans
from sklearn.metrics import (
    homogeneity_score,
    completeness_score,
    v_measure_score,
    adjusted_rand_score,
    adjusted_mutual_info_score,
)
def cluster_and_visualize(data, n_components, n_clusters=20, random_state=42):
    nmf_model = NMF(n_components=n_components, init='random', random_state=rando
    words_count_nmf = nmf_model.fit_transform(data)

    kmeans_model = KMeans(init='k-means++', max_iter=1000000, n_clusters=n_clust
    kmeans_model.fit(words_count_nmf)

    confusion_mat = confusion_matrix(news_dataset.target, kmeans_model.labels_)
    rows, cols = linear_sum_assignment(confusion_mat, maximize=True)
    plot_mat(confusion_mat[rows[:, np.newaxis], cols], xticklabels=cols, yticklа

    print("Homogeneity score for %s: %f" % ("", homogeneity_score(news_dataset.t
    print("Completeness score for %s: %f" % ("",  completeness_score(news_datase
    print("V-measure score for %s: %f" % ("",  v_measure_score(news_dataset.targ
    print("Adjusted Rand Index score for %s: %f" % ("",  adjusted_rand_score(new
    print("Adjusted mutual information score for %s: %f" % ("",  adjusted_mutual
```

```
# Usage
cluster_and_visualize(tfidf_dataframe, 10)
```




```
Homogeneity score for : 0.301093
Completeness score for : 0.346081
V-measure score for : 0.322023
Adjusted Rand Index score for : 0.101485
Adjusted mutual information score for : 0.319661
```

4. UMAP

Question 11

In [ ]:

```
!pip uninstall umap
!pip install umap-learn
!pip install umap-learn[plot]
```

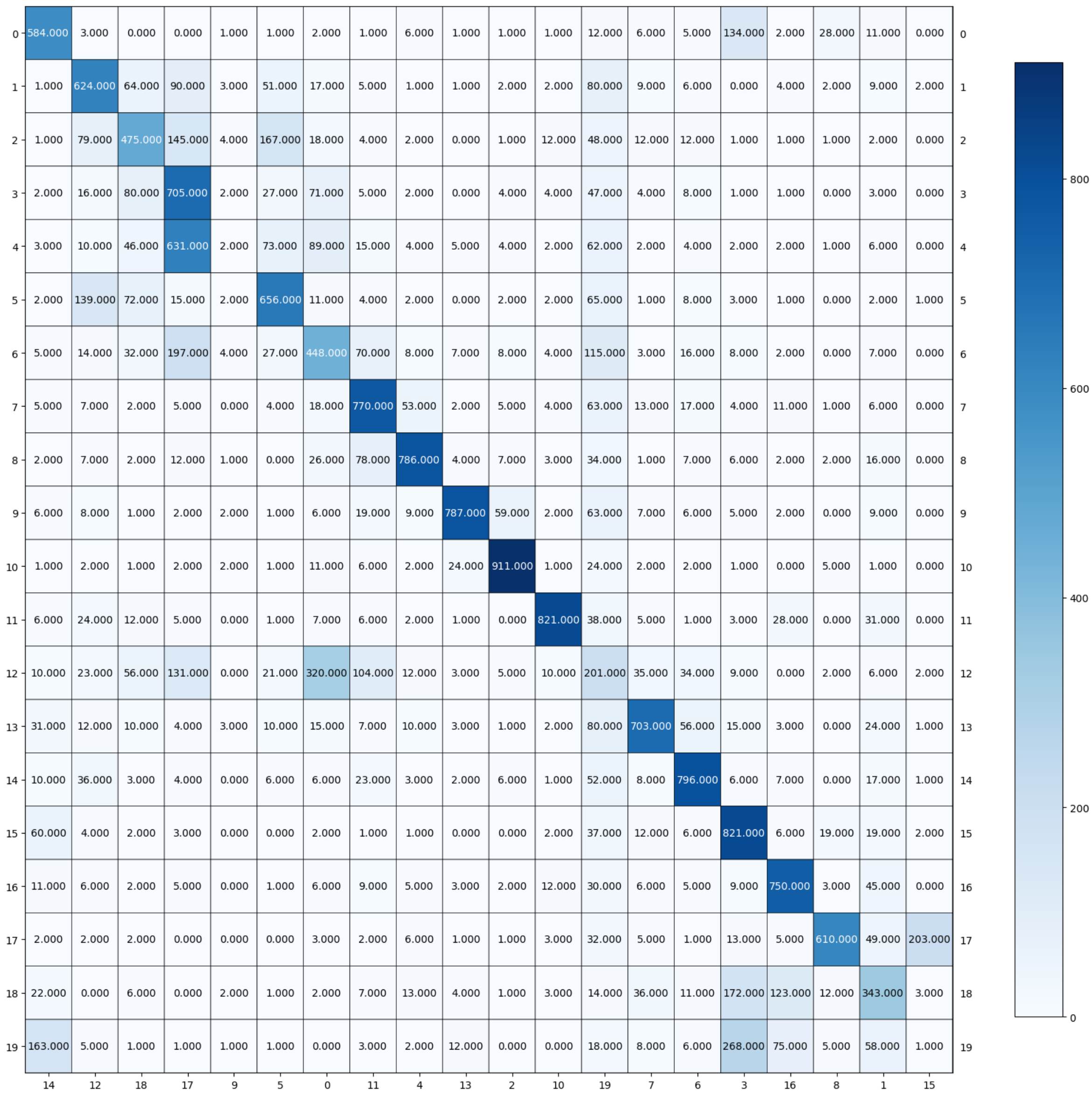
```
WARNING: Skipping umap as it is not installed.
Collecting umap-learn
  Downloading umap-learn-0.5.5.tar.gz (90 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 90.9/90.9 kB 3.4 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-pack
ages (from umap-learn) (1.23.5)
Requirement already satisfied: scipy>=1.3.1 in /usr/local/lib/python3.10/dist-pac
kages (from umap-learn) (1.11.4)
Requirement already satisfied: scikit-learn>=0.22 in /usr/local/lib/python3.10/di
st-packages (from umap-learn) (1.2.2)
Requirement already satisfied: numba>=0.51.2 in /usr/local/lib/python3.10/dist-pa
ckages (from umap-learn) (0.58.1)
Collecting pynndescent>=0.5 (from umap-learn)
  Downloading pynndescent-0.5.11-py3-none-any.whl (55 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 55.8/55.8 kB 5.7 MB/s eta 0:00:00
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (f
rom umap-learn) (4.66.1)
Requirement already satisfied: llvmlite<0.42,>=0.41.0dev0 in /usr/local/lib/pytho
n3.10/dist-packages (from numba>=0.51.2->umap-learn) (0.41.1)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.10/dist-pac
kages (from pynndescent>=0.5->umap-learn) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/
dist-packages (from scikit-learn>=0.22->umap-learn) (3.2.0)
Building wheels for collected packages: umap-learn
  Building wheel for umap-learn (setup.py) ... done
  Created wheel for umap-learn: filename=umap_learn-0.5.5-py3-none-any.whl size=8
6832 sha256=003673ab528446a0aa5477b9d9a77a5e70a2feffedb74182b62139d46f7ba3ca
  Stored in directory: /root/.cache/pip/wheels/3a/70/07/428d2b58660a1a3b431db59b8
06a10da736612ebbc66c1bcc5
Successfully built umap-learn
Installing collected packages: pynndescent, umap-learn
Successfully installed pynndescent-0.5.11 umap-learn-0.5.5
```

In [ ]:

```python
import umap.umap_ as umap
from sklearn.cluster import KMeans
from sklearn.metrics import confusion_matrix
from scipy.optimize import linear_sum_assignment

def run_umap_and_kmeans(tfidf_data, target_labels, distance_metric, n_components
    print(f'\nUMAP Results using {distance_metric} & n_components = {n_component
    umap_model = umap.UMAP(n_components=n_components, metric=distance_metric, ra
    umap_transformed = umap_model.fit_transform(tfidf_data)

    kmeans_clusterer = KMeans(random_state=0, n_clusters=n_clusters, max_iter=10
    kmeans_clusterer.fit(umap_transformed)

    print_cluster_metrics(target_labels, kmeans_clusterer.labels_)
    plot_confusion_matrix(target_labels, kmeans_clusterer.labels_)

def print_cluster_metrics(y_true, y_pred):
    print("Homogeneity score:", homogeneity_score(y_true, y_pred))
    print("Completeness score:", completeness_score(y_true, y_pred))
    print("V-measure score:", v_measure_score(y_true, y_pred))
    print("Adjusted Rand Index score:", adjusted_rand_score(y_true, y_pred))
    print("Adjusted Mutual Information score:", adjusted_mutual_info_score(y_tru

def plot_confusion_matrix(y_true, y_pred):
    cm = confusion_matrix(y_true, y_pred)
    rows, cols = linear_sum_assignment(cm, maximize=True)
```

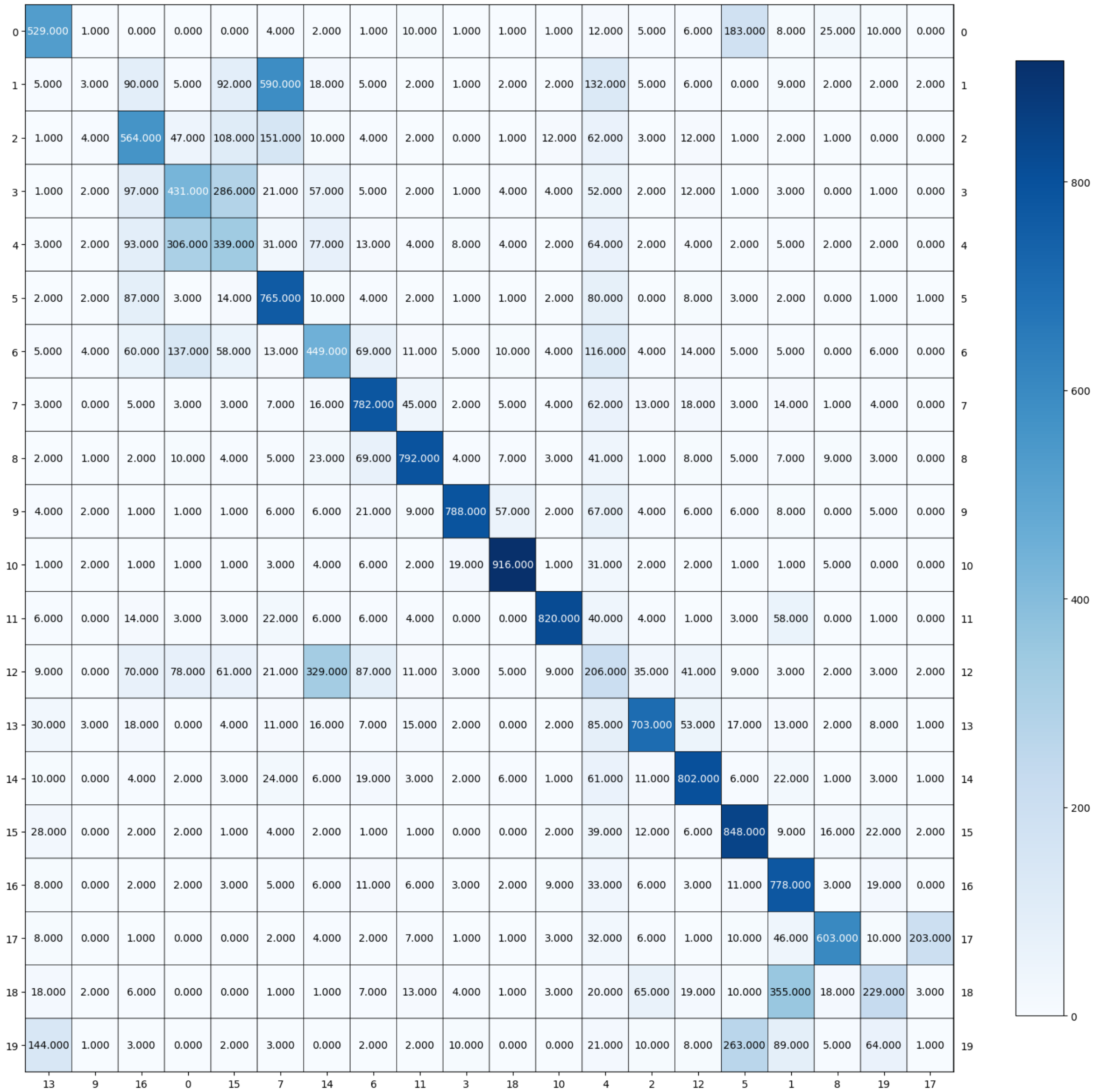
```
    plot_mat(cm[rows[:, np.newaxis], cols], xticklabels=cols, yticklabels=rows,

# Run UMAP for different parameters and metrics
umap_params_list = [(5, 'cosine'), (20, 'cosine'), (200, 'cosine'), (5, 'euclide
for n_components_value, distance_metric_value in umap_params_list:
    run_umap_and_kmeans(tfidf_dataframe, news_dataset.target, distance_metric_va
```

```
UMAP Results using cosine & n_components = 5:
/usr/local/lib/python3.10/dist-packages/umap/umap_.py:1943: UserWarning: n_jobs v
alue -1 overridden to 1 by setting random_state. Use no seed for parallelism.
  warn(f"n_jobs value {self.n_jobs} overridden to 1 by setting random_state. Use
no seed for parallelism.")
Homogeneity score: 0.5689113781606602
Completeness score: 0.58788157752841
V-measure score: 0.5782409320909574
Adjusted Rand Index score: 0.4524382375325882
Adjusted Mutual Information score: 0.5768448624199578
```

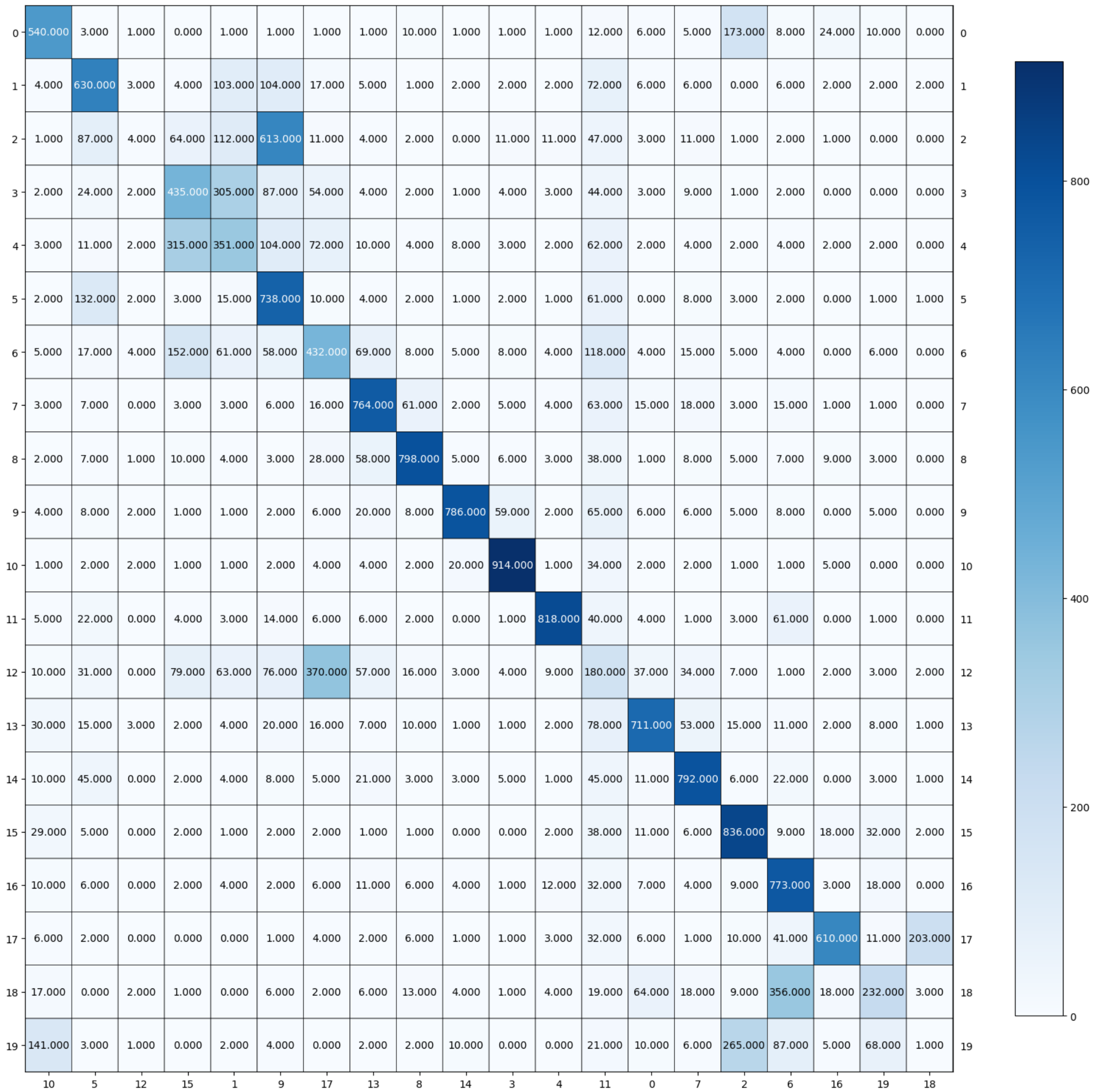



```
UMAP Results using cosine & n_components = 20:
/usr/local/lib/python3.10/dist-packages/umap/umap_.py:1943: UserWarning: n_jobs v
alue -1 overridden to 1 by setting random_state. Use no seed for parallelism.
  warn(f"n_jobs value {self.n_jobs} overridden to 1 by setting random_state. Use
no seed for parallelism.")
```

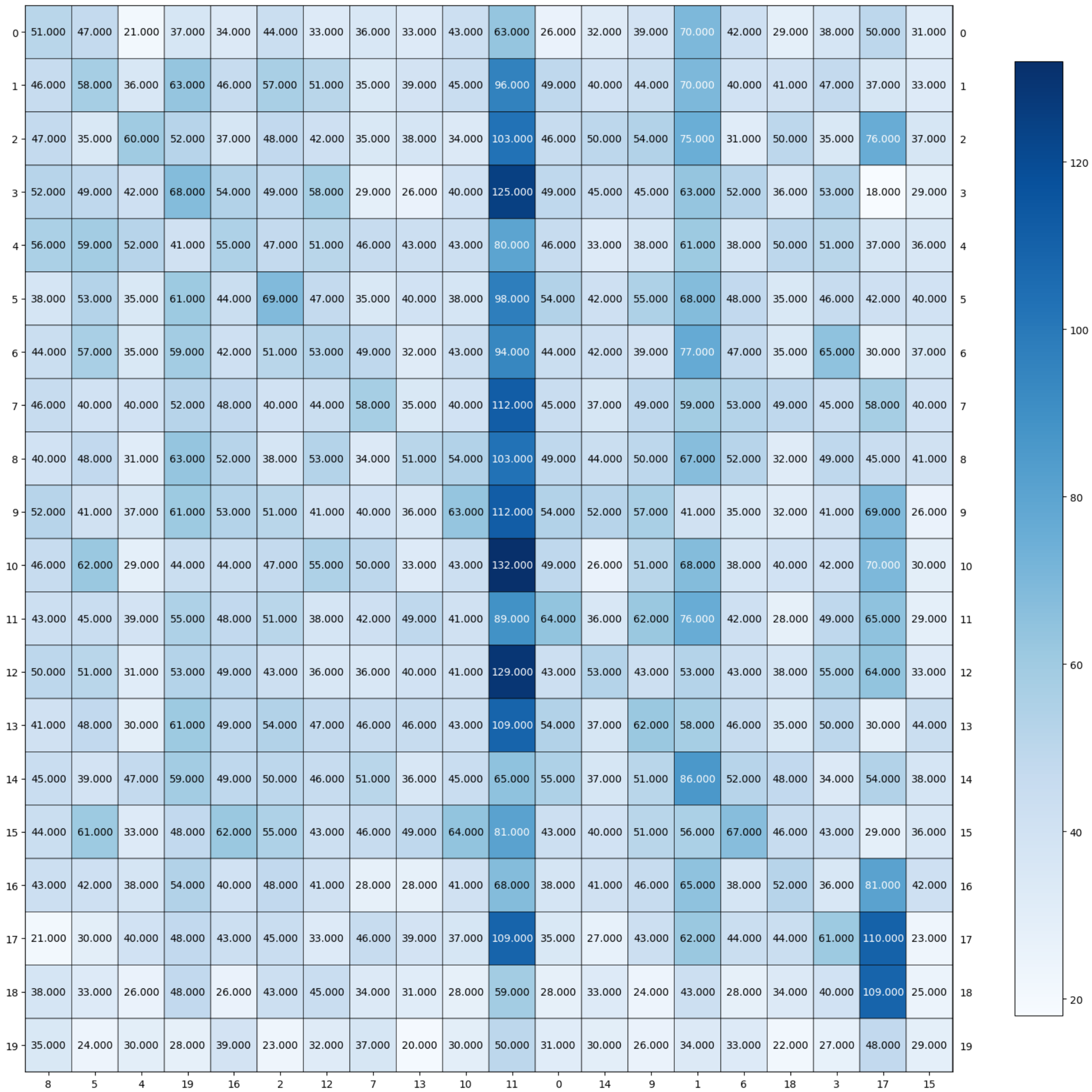
```
Homogeneity score: 0.5639415097627272
Completeness score: 0.5837578263288667
V-measure score: 0.5736785925777276
Adjusted Rand Index score: 0.44416336630818326
Adjusted Mutual Information score: 0.5722660045491073
```



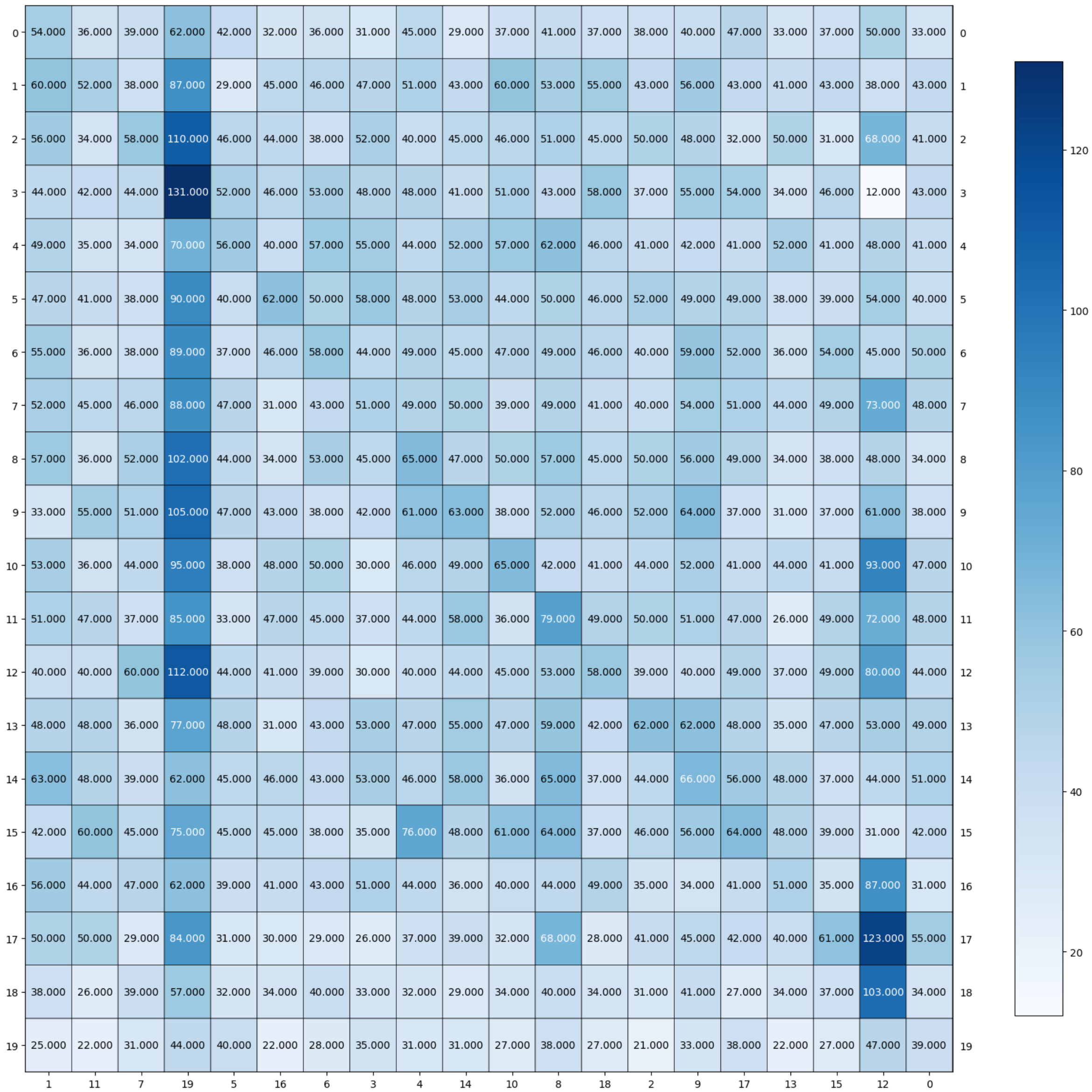

```
UMAP Results using cosine & n_components = 200:
/usr/local/lib/python3.10/dist-packages/umap/umap_.py:1943: UserWarning: n_jobs v
alue -1 overridden to 1 by setting random_state. Use no seed for parallelism.
  warn(f"n_jobs value {self.n_jobs} overridden to 1 by setting random_state. Use
no seed for parallelism.")
Homogeneity score: 0.5653100153929963
Completeness score: 0.5854613870627202
V-measure score: 0.5752092640226476
Adjusted Rand Index score: 0.44481434747002424
Adjusted Mutual Information score: 0.5738014235314004
```



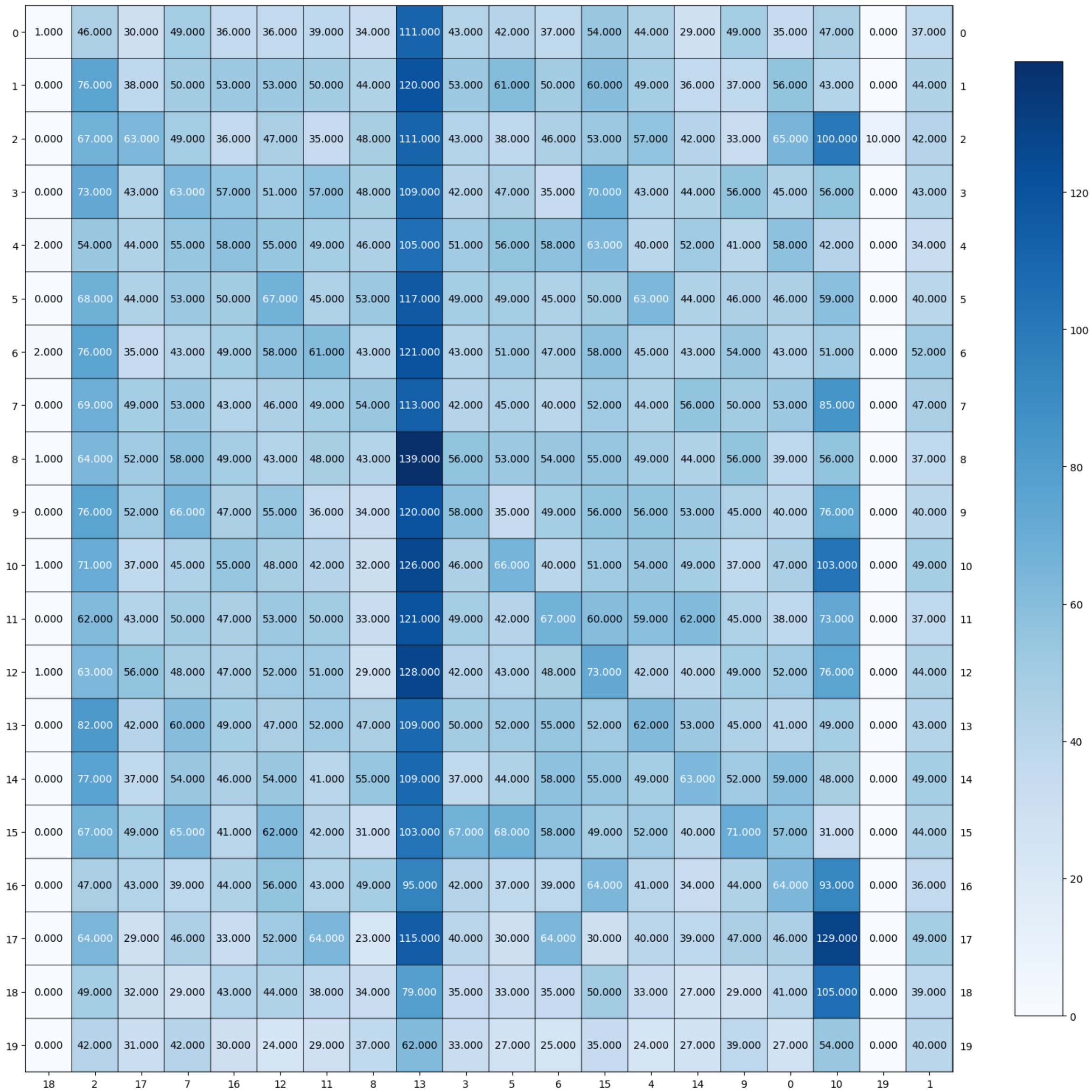

```
UMAP Results using euclidean & n_components = 5:
/usr/local/lib/python3.10/dist-packages/umap/umap_.py:1943: UserWarning: n_jobs v
alue -1 overridden to 1 by setting random_state. Use no seed for parallelism.
  warn(f"n_jobs value {self.n_jobs} overridden to 1 by setting random_state. Use
no seed for parallelism.")
Homogeneity score: 0.0067302586803309776
Completeness score: 0.006786437747219485
V-measure score: 0.006758231466033897
Adjusted Rand Index score: 0.0013768191959116626
Adjusted Mutual Information score: 0.003539773605795946
```




```
UMAP Results using euclidean & n_components = 20:
/usr/local/lib/python3.10/dist-packages/umap/umap_.py:1943: UserWarning: n_jobs v
alue -1 overridden to 1 by setting random_state. Use no seed for parallelism.
  warn(f"n_jobs value {self.n_jobs} overridden to 1 by setting random_state. Use
no seed for parallelism.")
Homogeneity score: 0.006631022103980117
Completeness score: 0.006661118524906376
V-measure score: 0.006646036241881955
Adjusted Rand Index score: 0.0013476182034869316
Adjusted Mutual Information score: 0.0034335506302220183
```




```
UMAP Results using euclidean & n_components = 200:
/usr/local/lib/python3.10/dist-packages/umap/umap_.py:1943: UserWarning: n_jobs v
alue -1 overridden to 1 by setting random_state. Use no seed for parallelism.
  warn(f"n_jobs value {self.n_jobs} overridden to 1 by setting random_state. Use
no seed for parallelism.")
Homogeneity score: 0.0061866039151025854
Completeness score: 0.006471681862863335
V-measure score: 0.006325932760940151
Adjusted Rand Index score: 0.000992228114740771
Adjusted Mutual Information score: 0.003051160084403079
```



Question 12

the first part highlights the superior performance of UMAP in dimensionality reduction, particularly when using the cosine metric. The robust diagonals in the contingency matrix and elevated metrics across various n_components emphasize its ability to achieve enhanced cluster separation compared to SVD/PCA and NMF. The selection of n_components=5, based on the highest V-score, is deemed optimal.

Conversely, the second part focuses on the suboptimal performance of UMAP dimensionality reduction with the Euclidean metric, indicated by low scores across homogeneity, completeness, v-measure, and adjusted random index for all n_component values. Despite its general unsuitability, the optimal setting is n_components=5, as observed in the confusion matrix, showing slightly improved cluster creation compared to other n_component values.

In conclusion, the cosine metric with n_components=5 remains the preferred choice for UMAP.

Question 13

In [7]:
```
from sklearn.cluster import KMeans
from sklearn.metrics import cluster
kmeans = KMeans(random_state=0, n_clusters=20, max_iter=1000, n_init=30)
kmeans.fit(tfidf_dataframe)
```

Out[7]:
```
▾ KMeans
KMeans(max_iter=1000, n_clusters=20, n_init=30, random_state=0)
```

In [8]:
```
print("Homogeneity: ", cluster.homogeneity_score(news_dataset.target, kmeans.lab
print("Completeness: ",cluster. completeness_score(news_dataset.target, kmeans.l
print("V-measure: ", cluster.v_measure_score(news_dataset.target, kmeans.labels_
print("Adjusted Rand-Index: ", cluster.adjusted_rand_score(news_dataset.target,
print("Adjusted Mutual Information Score: ", cluster.adjusted_mutual_info_score(
```

```
Homogeneity:  0.326807011612208
Completeness:  0.3743410597965642
V-measure:  0.3489627599775251
Adjusted Rand-Index:  0.11489276920106191
Adjusted Mutual Information Score:  0.3467089692894358
```

Clustering Algorithms that do not explicitly rely on the Gaussian distribution per cluster

1. Agglomerative Clustering

Question 14

In [ ]:

In [ ]:

In [ ]:

2. HDBSCAN

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Part 2

```
In [ ]:
```