main

April 8, 2023

1 Detecting Communities in the Reddit Hyperlink Dataset

1.1 For ECMM447 - Social Networks and Text Analysis

[34]:
```
# Only run once to install necessary dependencies
!pip install scikit-image
!pip install python-louvain
!pip install cdlib
```

```
Requirement already satisfied: scikit-image in c:\users\aorun\desktop\uni\year
4\term 2\ecmm447 - social networks and text
analysis\ca\redditcommunitydetection\.venv\lib\site-packages (0.20.0)
Requirement already satisfied: packaging>=20.0 in
c:\users\aorun\desktop\uni\year 4\term 2\ecmm447 - social networks and text
analysis\ca\redditcommunitydetection\.venv\lib\site-packages (from scikit-image)
(23.0)
Requirement already satisfied: scipy<1.9.2,>=1.8 in
c:\users\aorun\desktop\uni\year 4\term 2\ecmm447 - social networks and text
analysis\ca\redditcommunitydetection\.venv\lib\site-packages (from scikit-image)
(1.9.1)
Requirement already satisfied: imageio>=2.4.1 in c:\users\aorun\desktop\uni\year
4\term 2\ecmm447 - social networks and text
analysis\ca\redditcommunitydetection\.venv\lib\site-packages (from scikit-image)
(2.27.0)
Requirement already satisfied: pillow>=9.0.1 in c:\users\aorun\desktop\uni\year
4\term 2\ecmm447 - social networks and text
analysis\ca\redditcommunitydetection\.venv\lib\site-packages (from scikit-image)
(9.5.0)
Requirement already satisfied: tifffile>=2019.7.26 in
c:\users\aorun\desktop\uni\year 4\term 2\ecmm447 - social networks and text
analysis\ca\redditcommunitydetection\.venv\lib\site-packages (from scikit-image)
(2023.3.21)
Requirement already satisfied: PyWavelets>=1.1.1 in
c:\users\aorun\desktop\uni\year 4\term 2\ecmm447 - social networks and text
analysis\ca\redditcommunitydetection\.venv\lib\site-packages (from scikit-image)
(1.4.1)
Requirement already satisfied: numpy>=1.21.1 in c:\users\aorun\desktop\uni\year
4\term 2\ecmm447 - social networks and text
```

```
analysis\ca\redditcommunitydetection\.venv\lib\site-packages (from scikit-image)
(1.23.5)
Requirement already satisfied: lazy_loader>=0.1 in
c:\users\aorun\desktop\uni\year 4\term 2\ecmm447 - social networks and text
analysis\ca\redditcommunitydetection\.venv\lib\site-packages (from scikit-image)
(0.2)
Requirement already satisfied: networkx>=2.8 in c:\users\aorun\desktop\uni\year
4\term 2\ecmm447 - social networks and text
analysis\ca\redditcommunitydetection\.venv\lib\site-packages (from scikit-image)
(3.0)
Requirement already satisfied: python-louvain in c:\users\aorun\desktop\uni\year
4\term 2\ecmm447 - social networks and text
analysis\ca\redditcommunitydetection\.venv\lib\site-packages (0.16)
Requirement already satisfied: numpy in c:\users\aorun\desktop\uni\year 4\term
2\ecmm447 - social networks and text
analysis\ca\redditcommunitydetection\.venv\lib\site-packages (from python-
louvain) (1.23.5)
Requirement already satisfied: networkx in c:\users\aorun\desktop\uni\year
4\term 2\ecmm447 - social networks and text
analysis\ca\redditcommunitydetection\.venv\lib\site-packages (from python-
louvain) (3.0)
Collecting cdlib
  Downloading cdlib-0.2.6-py3-none-any.whl (228 kB)
     ---------------------------------------- 0.0/228.6 kB ? eta -:--:--
     -------------------------------- ----- 194.6/228.6 kB 5.9 MB/s eta 0:00:01
     ---------------------------------------- 228.6/228.6 kB 7.0 MB/s eta 0:00:00
Collecting future
  Downloading future-0.18.3.tar.gz (840 kB)
     ---------------------------------------- 0.0/840.9 kB ? eta -:--:--
     ---------- -------------------------- 225.3/840.9 kB 4.6 MB/s eta 0:00:01
     ------------------ -------------------- 409.6/840.9 kB 5.1 MB/s eta 0:00:01
     ---------------------- --------------- 491.5/840.9 kB 4.4 MB/s eta 0:00:01
     -------------------------------- ----- 727.0/840.9 kB 5.1 MB/s eta 0:00:01
     ---------------------------------------- 840.9/840.9 kB 4.8 MB/s eta 0:00:00
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Requirement already satisfied: matplotlib in c:\users\aorun\desktop\uni\year
4\term 2\ecmm447 - social networks and text
analysis\ca\redditcommunitydetection\.venv\lib\site-packages (from cdlib)
(3.7.1)
Collecting python-Levenshtein
  Downloading python_Levenshtein-0.20.9-py3-none-any.whl (9.4 kB)
Requirement already satisfied: scipy in c:\users\aorun\desktop\uni\year 4\term
2\ecmm447 - social networks and text
analysis\ca\redditcommunitydetection\.venv\lib\site-packages (from cdlib)
(1.9.1)
Collecting bimlpa
  Downloading bimlpa-0.1.2-py3-none-any.whl (7.0 kB)
```

```
Collecting eva-lcd
  Downloading eva_lcd-0.1.1-py3-none-any.whl (9.2 kB)
Collecting chinese-whispers
  Downloading chinese_whispers-0.8.1-py3-none-any.whl (7.8 kB)
Collecting pyclustering
  Downloading pyclustering-0.10.1.2.tar.gz (2.6 MB)
     ---------------------------------------- 0.0/2.6 MB ? eta -:--:--
     --- ------------------------------------ 0.2/2.6 MB 6.1 MB/s eta 0:00:01
     ------ ---------------------------------- 0.4/2.6 MB 5.5 MB/s eta 0:00:01
     --------- ------------------------------- 0.6/2.6 MB 5.4 MB/s eta 0:00:01
     ------------ -------------------------- 0.8/2.6 MB 5.5 MB/s eta 0:00:01
     --------------- ------------------------ 0.9/2.6 MB 4.8 MB/s eta 0:00:01
     ----------------- ---------------------- 1.1/2.6 MB 4.9 MB/s eta 0:00:01
     -------------------- ------------------- 1.3/2.6 MB 5.3 MB/s eta 0:00:01
     ----------------------- ---------------- 1.5/2.6 MB 5.4 MB/s eta 0:00:01
     ------------------------ --------------- 1.6/2.6 MB 5.2 MB/s eta 0:00:01
     --------------------------- ------------ 1.7/2.6 MB 4.7 MB/s eta 0:00:01
     ----------------------------- ---------- 1.9/2.6 MB 4.9 MB/s eta 0:00:01
     ------------------------------- -------- 2.1/2.6 MB 4.9 MB/s eta 0:00:01
     ---------------------------------- ----- 2.3/2.6 MB 4.9 MB/s eta 0:00:01
     ------------------------------------  2.5/2.6 MB 5.0 MB/s eta 0:00:01
     ---------------------------------------- 2.6/2.6 MB 5.0 MB/s eta 0:00:00
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Collecting dynetx
  Downloading dynetx-0.3.1-py3-none-any.whl (39 kB)
Collecting pooch
  Downloading pooch-1.7.0-py3-none-any.whl (60 kB)
     ---------------------------------------- 0.0/60.9 kB ? eta -:--:--
     ---------------------------------------- 60.9/60.9 kB ? eta 0:00:00
Collecting cython
  Downloading Cython-0.29.34-py2.py3-none-any.whl (988 kB)
     ---------------------------------------- 0.0/988.1 kB ? eta -:--:--
     ---------- ----------------------------- 266.2/988.1 kB 8.3 MB/s eta 0:00:01
     ----------------- ---------------------- 481.3/988.1 kB 7.5 MB/s eta 0:00:01
     ------------------------ ------------ 675.8/988.1 kB 7.1 MB/s eta 0:00:01
     ---------------------------------- -- 911.4/988.1 kB 6.4 MB/s eta 0:00:01
     ---------------------------------------- 988.1/988.1 kB 6.9 MB/s eta 0:00:00
Collecting pulp
  Using cached PuLP-2.7.0-py3-none-any.whl (14.3 MB)
Collecting thresholdclustering
  Downloading thresholdclustering-1.1-py3-none-any.whl (5.3 kB)
Requirement already satisfied: networkx>=2.4 in c:\users\aorun\desktop\uni\year
4\term 2\ecmm447 - social networks and text
analysis\ca\redditcommunitydetection\.venv\lib\site-packages (from cdlib) (3.0)
Collecting tqdm
  Downloading tqdm-4.65.0-py3-none-any.whl (77 kB)
     ---------------------------------------- 0.0/77.1 kB ? eta -:--:--
```

```
     ---------------------------------------- 77.1/77.1 kB 4.2 MB/s eta 0:00:00
Collecting demon
  Downloading demon-2.0.6-py3-none-any.whl (7.3 kB)
Collecting angel-cd
  Downloading angel_cd-1.0.3-py3-none-any.whl (10 kB)
Collecting seaborn
  Using cached seaborn-0.12.2-py3-none-any.whl (293 kB)
Collecting python-igraph
  Downloading python-igraph-0.10.4.tar.gz (9.5 kB)
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Collecting nf1
  Downloading nf1-0.0.4-py3-none-any.whl (18 kB)
Requirement already satisfied: pandas in c:\users\aorun\desktop\uni\year 4\term
2\ecmm447 - social networks and text
analysis\ca\redditcommunitydetection\.venv\lib\site-packages (from cdlib)
(1.5.3)
Requirement already satisfied: numpy in c:\users\aorun\desktop\uni\year 4\term
2\ecmm447 - social networks and text
analysis\ca\redditcommunitydetection\.venv\lib\site-packages (from cdlib)
(1.23.5)
Requirement already satisfied: python-louvain>=0.16 in
c:\users\aorun\desktop\uni\year 4\term 2\ecmm447 - social networks and text
analysis\ca\redditcommunitydetection\.venv\lib\site-packages (from cdlib) (0.16)
Collecting scikit-learn
  Downloading scikit_learn-1.2.2-cp39-cp39-win_amd64.whl (8.4 MB)
     ---------------------------------------- 0.0/8.4 MB ? eta -:--:--
     - -------------------------------------- 0.2/8.4 MB 7.6 MB/s eta 0:00:02
     - -------------------------------------- 0.3/8.4 MB 7.0 MB/s eta 0:00:02
     -- ------------------------------------- 0.5/8.4 MB 4.7 MB/s eta 0:00:02
     -- ------------------------------------- 0.6/8.4 MB 4.6 MB/s eta 0:00:02
     -- ------------------------------------- 0.6/8.4 MB 3.5 MB/s eta 0:00:03
     --- ------------------------------------ 0.8/8.4 MB 3.9 MB/s eta 0:00:02
     ----- ---------------------------------- 1.1/8.4 MB 4.2 MB/s eta 0:00:02
     ----- ---------------------------------- 1.1/8.4 MB 4.3 MB/s eta 0:00:02
     ----- ---------------------------------- 1.2/8.4 MB 4.0 MB/s eta 0:00:02
     ------ --------------------------------- 1.4/8.4 MB 4.2 MB/s eta 0:00:02
     -------- ------------------------------- 1.7/8.4 MB 4.3 MB/s eta 0:00:02
     --------- ------------------------------ 1.9/8.4 MB 4.6 MB/s eta 0:00:02
     ---------- ----------------------------- 2.2/8.4 MB 4.8 MB/s eta 0:00:02
     ---------- ----------------------------- 2.3/8.4 MB 4.7 MB/s eta 0:00:02
     ----------- ---------------------------- 2.5/8.4 MB 4.7 MB/s eta 0:00:02
     ------------- -------------------------- 2.7/8.4 MB 4.8 MB/s eta 0:00:02
     -------------- ------------------------- 3.0/8.4 MB 5.0 MB/s eta 0:00:02
     --------------- ------------------------ 3.2/8.4 MB 5.1 MB/s eta 0:00:02
     --------------- ------------------------ 3.3/8.4 MB 5.0 MB/s eta 0:00:02
     ---------------- ----------------------- 3.5/8.4 MB 4.9 MB/s eta 0:00:01
     ----------------- ---------------------- 3.7/8.4 MB 5.0 MB/s eta 0:00:01
```

```
     ----------------- ---------------------- 3.8/8.4 MB 5.0 MB/s eta 0:00:01
     ----------------- ---------------------- 4.0/8.4 MB 4.9 MB/s eta 0:00:01
     ------------------ --------------------- 4.1/8.4 MB 4.8 MB/s eta 0:00:01
     ------------------ --------------------- 4.1/8.4 MB 4.8 MB/s eta 0:00:01
     ------------------- -------------------- 4.2/8.4 MB 4.6 MB/s eta 0:00:01
     ------------------- -------------------- 4.4/8.4 MB 4.6 MB/s eta 0:00:01
     -------------------- ------------------- 4.5/8.4 MB 4.6 MB/s eta 0:00:01
     --------------------- ------------------ 4.6/8.4 MB 4.6 MB/s eta 0:00:01
     ---------------------- ----------------- 4.9/8.4 MB 4.6 MB/s eta 0:00:01
     ----------------------- ---------------- 5.1/8.4 MB 4.7 MB/s eta 0:00:01
     ------------------------ --------------- 5.3/8.4 MB 4.7 MB/s eta 0:00:01
     ------------------------- -------------- 5.6/8.4 MB 4.8 MB/s eta 0:00:01
     -------------------------- ------------- 5.8/8.4 MB 4.8 MB/s eta 0:00:01
     --------------------------- ------------ 5.9/8.4 MB 4.8 MB/s eta 0:00:01
     ---------------------------- ----------- 6.2/8.4 MB 4.8 MB/s eta 0:00:01
     ----------------------------- ---------- 6.5/8.4 MB 4.9 MB/s eta 0:00:01
     ------------------------------ --------- 6.7/8.4 MB 5.0 MB/s eta 0:00:01
     ------------------------------- -------- 6.8/8.4 MB 5.0 MB/s eta 0:00:01
     -------------------------------- ------- 6.9/8.4 MB 4.9 MB/s eta 0:00:01
     --------------------------------- ------ 7.2/8.4 MB 4.9 MB/s eta 0:00:01
     ---------------------------------- ----- 7.4/8.4 MB 5.0 MB/s eta 0:00:01
     ----------------------------------- ---- 7.7/8.4 MB 5.0 MB/s eta 0:00:01
     ------------------------------------ --- 7.9/8.4 MB 5.0 MB/s eta 0:00:01
     ------------------------------------- -- 8.1/8.4 MB 5.0 MB/s eta 0:00:01
     -------------------------------------  8.3/8.4 MB 5.1 MB/s eta 0:00:01
     -------------------------------------- 8.4/8.4 MB 5.0 MB/s eta 0:00:00
Collecting markov-clustering
  Downloading markov_clustering-0.0.6.dev0-py3-none-any.whl (6.3 kB)
Collecting networkx>=2.4
  Using cached networkx-2.8.8-py3-none-any.whl (2.0 MB)
Requirement already satisfied: decorator in c:\users\aorun\desktop\uni\year
4\term 2\ecmm447 - social networks and text
analysis\ca\redditcommunitydetection\.venv\lib\site-packages (from
dynetx->cdlib) (5.1.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
c:\users\aorun\desktop\uni\year 4\term 2\ecmm447 - social networks and text
analysis\ca\redditcommunitydetection\.venv\lib\site-packages (from
matplotlib->cdlib) (1.4.4)
Requirement already satisfied: pyparsing>=2.3.1 in
c:\users\aorun\desktop\uni\year 4\term 2\ecmm447 - social networks and text
analysis\ca\redditcommunitydetection\.venv\lib\site-packages (from
matplotlib->cdlib) (3.0.9)
Requirement already satisfied: fonttools>=4.22.0 in
c:\users\aorun\desktop\uni\year 4\term 2\ecmm447 - social networks and text
analysis\ca\redditcommunitydetection\.venv\lib\site-packages (from
matplotlib->cdlib) (4.39.3)
Requirement already satisfied: packaging>=20.0 in
c:\users\aorun\desktop\uni\year 4\term 2\ecmm447 - social networks and text
```

```
analysis\ca\redditcommunitydetection\.venv\lib\site-packages (from
matplotlib->cdlib) (23.0)
Requirement already satisfied: python-dateutil>=2.7 in
c:\users\aorun\desktop\uni\year 4\term 2\ecmm447 - social networks and text
analysis\ca\redditcommunitydetection\.venv\lib\site-packages (from
matplotlib->cdlib) (2.8.2)
Requirement already satisfied: contourpy>=1.0.1 in
c:\users\aorun\desktop\uni\year 4\term 2\ecmm447 - social networks and text
analysis\ca\redditcommunitydetection\.venv\lib\site-packages (from
matplotlib->cdlib) (1.0.7)
Requirement already satisfied: importlib-resources>=3.2.0 in
c:\users\aorun\desktop\uni\year 4\term 2\ecmm447 - social networks and text
analysis\ca\redditcommunitydetection\.venv\lib\site-packages (from
matplotlib->cdlib) (5.12.0)
Requirement already satisfied: cycler>=0.10 in c:\users\aorun\desktop\uni\year
4\term 2\ecmm447 - social networks and text
analysis\ca\redditcommunitydetection\.venv\lib\site-packages (from
matplotlib->cdlib) (0.11.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\aorun\desktop\uni\year
4\term 2\ecmm447 - social networks and text
analysis\ca\redditcommunitydetection\.venv\lib\site-packages (from
matplotlib->cdlib) (9.5.0)
Requirement already satisfied: pytz>=2020.1 in c:\users\aorun\desktop\uni\year
4\term 2\ecmm447 - social networks and text
analysis\ca\redditcommunitydetection\.venv\lib\site-packages (from
pandas->cdlib) (2023.3)
Requirement already satisfied: platformdirs>=2.5.0 in
c:\users\aorun\desktop\uni\year 4\term 2\ecmm447 - social networks and text
analysis\ca\redditcommunitydetection\.venv\lib\site-packages (from pooch->cdlib)
(3.2.0)
Requirement already satisfied: requests>=2.19.0 in
c:\users\aorun\desktop\uni\year 4\term 2\ecmm447 - social networks and text
analysis\ca\redditcommunitydetection\.venv\lib\site-packages (from pooch->cdlib)
(2.28.2)
Requirement already satisfied: igraph==0.10.4 in c:\users\aorun\desktop\uni\year
4\term 2\ecmm447 - social networks and text
analysis\ca\redditcommunitydetection\.venv\lib\site-packages (from python-
igraph->cdlib) (0.10.4)
Requirement already satisfied: texttable>=1.6.2 in
c:\users\aorun\desktop\uni\year 4\term 2\ecmm447 - social networks and text
analysis\ca\redditcommunitydetection\.venv\lib\site-packages (from
igraph==0.10.4->python-igraph->cdlib) (1.6.7)
Collecting Levenshtein==0.20.9
  Downloading Levenshtein-0.20.9-cp39-cp39-win_amd64.whl (101 kB)
     ---------------------------------------- 0.0/101.3 kB ? eta -:--:--
     ---------------------- --------------- 61.4/101.3 kB 1.6 MB/s eta 0:00:01
     ---------------------------------------- 101.3/101.3 kB 1.9 MB/s eta 0:00:00
Collecting rapidfuzz<3.0.0,>=2.3.0
```

```
  Downloading rapidfuzz-2.15.0-cp39-cp39-win_amd64.whl (1.1 MB)
     ---------------------------------------- 0.0/1.1 MB ? eta -:--:--
     -------- ------------------------------- 0.2/1.1 MB 6.9 MB/s eta 0:00:01
     ---------------- ----------------------- 0.5/1.1 MB 5.6 MB/s eta 0:00:01
     ------------------------- -------------- 0.7/1.1 MB 6.3 MB/s eta 0:00:01
     --------------------------------- ------ 0.9/1.1 MB 6.3 MB/s eta 0:00:01
     ---------------------------------------- 1.1/1.1 MB 6.1 MB/s eta 0:00:00
Collecting threadpoolctl>=2.0.0
  Using cached threadpoolctl-3.1.0-py3-none-any.whl (14 kB)
Collecting joblib>=1.1.1
  Using cached joblib-1.2.0-py3-none-any.whl (297 kB)
Requirement already satisfied: colorama in c:\users\aorun\desktop\uni\year
4\term 2\ecmm447 - social networks and text
analysis\ca\redditcommunitydetection\.venv\lib\site-packages (from tqdm->cdlib)
(0.4.6)
Requirement already satisfied: zipp>=3.1.0 in c:\users\aorun\desktop\uni\year
4\term 2\ecmm447 - social networks and text
analysis\ca\redditcommunitydetection\.venv\lib\site-packages (from importlib-
resources>=3.2.0->matplotlib->cdlib) (3.15.0)
Requirement already satisfied: six>=1.5 in c:\users\aorun\desktop\uni\year
4\term 2\ecmm447 - social networks and text
analysis\ca\redditcommunitydetection\.venv\lib\site-packages (from python-
dateutil>=2.7->matplotlib->cdlib) (1.16.0)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
c:\users\aorun\desktop\uni\year 4\term 2\ecmm447 - social networks and text
analysis\ca\redditcommunitydetection\.venv\lib\site-packages (from
requests>=2.19.0->pooch->cdlib) (1.26.15)
Requirement already satisfied: certifi>=2017.4.17 in
c:\users\aorun\desktop\uni\year 4\term 2\ecmm447 - social networks and text
analysis\ca\redditcommunitydetection\.venv\lib\site-packages (from
requests>=2.19.0->pooch->cdlib) (2022.12.7)
Requirement already satisfied: idna<4,>=2.5 in c:\users\aorun\desktop\uni\year
4\term 2\ecmm447 - social networks and text
analysis\ca\redditcommunitydetection\.venv\lib\site-packages (from
requests>=2.19.0->pooch->cdlib) (3.4)
Requirement already satisfied: charset-normalizer<4,>=2 in
c:\users\aorun\desktop\uni\year 4\term 2\ecmm447 - social networks and text
analysis\ca\redditcommunitydetection\.venv\lib\site-packages (from
requests>=2.19.0->pooch->cdlib) (3.1.0)
Building wheels for collected packages: future, pyclustering, python-igraph
  Building wheel for future (setup.py): started
  Building wheel for future (setup.py): finished with status 'done'
  Created wheel for future: filename=future-0.18.3-py3-none-any.whl size=492055
sha256=17fa293dbc19435c1ddb090b392fd0f4e90b4671366cb347a88f9af0eb4e0382
  Stored in directory: c:\users\aorun\appdata\local\pip\cache\wheels\bf\5d\6a\2e
53874f7ec4e2bede522385439531fafec8fafe005b5c3d1b
  Building wheel for pyclustering (setup.py): started
  Building wheel for pyclustering (setup.py): finished with status 'done'
```

```
  Created wheel for pyclustering: filename=pyclustering-0.10.1.2-py3-none-
any.whl size=2395137
sha256=f939ca089513971b0b951471cfc8b05510d18b36a1de0da0dc83745df61b52ed
  Stored in directory: c:\users\aorun\appdata\local\pip\cache\wheels\e0\56\c2\ab
b6866a3fcd8a55862f1df8a18f57805c3a78fed9a9023cb9
  Building wheel for python-igraph (setup.py): started
  Building wheel for python-igraph (setup.py): finished with status 'done'
  Created wheel for python-igraph: filename=python_igraph-0.10.4-py3-none-
any.whl size=9092
sha256=daeefa2b7e9c9e7512277247089cbfb9103e52bad2362a56d4aca8ea73614267
  Stored in directory: c:\users\aorun\appdata\local\pip\cache\wheels\dc\07\ac\bf
f79052fd6222d1239b228cd24a47222f227c2350f9c4df01
Successfully built future pyclustering python-igraph
Installing collected packages: pulp, tqdm, threadpoolctl, rapidfuzz, networkx,
joblib, future, cython, thresholdclustering, scikit-learn, python-igraph, pooch,
Levenshtein, eva-lcd, dynetx, demon, chinese-whispers, seaborn, python-
Levenshtein, pyclustering, nf1, markov-clustering, bimlpa, angel-cd, cdlib
  Attempting uninstall: networkx
    Found existing installation: networkx 3.0
    Uninstalling networkx-3.0:
      Successfully uninstalled networkx-3.0
Successfully installed Levenshtein-0.20.9 angel-cd-1.0.3 bimlpa-0.1.2
cdlib-0.2.6 chinese-whispers-0.8.1 cython-0.29.34 demon-2.0.6 dynetx-0.3.1 eva-
lcd-0.1.1 future-0.18.3 joblib-1.2.0 markov-clustering-0.0.6.dev0 networkx-2.8.8
nf1-0.0.4 pooch-1.7.0 pulp-2.7.0 pyclustering-0.10.1.2 python-Levenshtein-0.20.9
python-igraph-0.10.4 rapidfuzz-2.15.0 scikit-learn-1.2.2 seaborn-0.12.2
threadpoolctl-3.1.0 thresholdclustering-1.1 tqdm-4.65.0
```

[47]:
```
import numpy as np
from collections import Counter
import time

# Pandas Imports
import pandas
import pandas as pd

# Matplotlib Imports
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors
%matplotlib inline

# NetworkX Imports
import networkx as nx
from networkx.algorithms import community as comm

# Louvain Community Detection
import community
```

```
# Iteration Tools for Girvan-Newman
from itertools import islice, takewhile

# Imports for Downloading TSV
import urllib.request as req
import os.path as path

# CDLib Imports for Community Detection
from cdlib import NodeClustering
from cdlib.evaluation import triangle_participation_ratio, conductance,␣
 ↪newman_girvan_modularity

# Datashader Imports
import colorcet as cc
import datashader as ds
import datashader.transfer_functions as tf
from datashader.layout import circular_layout, forceatlas2_layout
from datashader.bundling import directly_connect_edges, hammer_bundle
from datashader.utils import export_image
```

[4]:
```
# Downloads the Reddit Hyperlink dataset TSV file and saves it to ./data/
url = 'https://snap.stanford.edu/data/soc-redditHyperlinks-body.tsv'
body_save = "./data/soc-redditHyperlinks-body.tsv"
print("Checking for 'soc-redditHyperlinks-body.tsv'...")
if not path.exists(body_save):
    print("File not Found.")
    print(f"Downloading 'soc-redditHyperlinks-body.tsv' from URL: '{url}'...")
    req.urlretrieve(url, body_save)
else:
    print(f"Found at Path: '{body_save}'!")

url = 'https://snap.stanford.edu/data/soc-redditHyperlinks-title.tsv'
title_save = "./data/soc-redditHyperlinks-title.tsv"
print("Checking for 'soc-redditHyperlinks-title.tsv'...")
if not path.exists(title_save):
    print("File not Found.")
    print(f"Downloading 'soc-redditHyperlinks-title.tsv' from URL: '{url}'...")
    req.urlretrieve(url, title_save)
    print("Done!")
else:
    print(f"Found at Path: '{title_save}'!")

title_path = title_save
body_path = body_save

# Load the TSV files and convert to CSV files
```

```
title_df = pd.read_table(title_path, sep="\t")
title_csv = title_path[:-4] + ".csv"
print("Checking for 'soc-redditHyperlinks-title.csv'...")
if not path.exists(title_csv):
    print(f"Converting TSV to CSV file at: '{title_csv}'...")
    title_df.to_csv(title_csv)
    print("Converted!")
else:
    print(f"Found at Path: {title_csv}")

body_df = pd.read_table(body_path, sep='\t')
body_csv = body_path[:-4] + ".csv"
print("Checking for 'soc-redditHyperlinks-body.csv'...")
if not path.exists(body_csv):
    print(f"Converting TSV to CSV file at: '{body_csv}'...")
    body_df.to_csv(body_csv)
    print("Converted!")
else:
    print(f"Found at Path: {body_csv}")

# Load CSV files into DataFrames, concatenate them and extract Source/Target␣
 ↪nodes
body_df = pd.read_csv(body_csv)
title_df = pd.read_csv(title_csv)

print("Creating edgelist DataFrame...")
reddit_df = pd.concat([title_df, body_df]).reset_index(drop=True)
reddit_df = reddit_df[["SOURCE_SUBREDDIT", "TARGET_SUBREDDIT"]]

reddit_df = reddit_df.groupby(["SOURCE_SUBREDDIT", "TARGET_SUBREDDIT"]).size().
 ↪reset_index(name='WEIGHT')
print("Done!")

reddit_path = "./data/redditHyperlinks-subredditsOnly.csv"
print(f"Saving edge details to CSV file at: '{reddit_path}'...")
reddit_df.to_csv(reddit_path, index=False)
print("Saved!")
```

```
Checking for 'soc-redditHyperlinks-body.tsv'…
Found at Path: './data/soc-redditHyperlinks-body.tsv'!
Checking for 'soc-redditHyperlinks-title.tsv'…
Found at Path: './data/soc-redditHyperlinks-title.tsv'!
Checking for 'soc-redditHyperlinks-title.csv'…
Found at Path: ./data/soc-redditHyperlinks-title.csv
Checking for 'soc-redditHyperlinks-body.csv'…
Found at Path: ./data/soc-redditHyperlinks-body.csv
Creating edgelist DataFrame…
```

```
Done!
Saving edge details to CSV file at: './data/redditHyperlinks-
subredditsOnly.csv'…
Saved!
```

The following function loads a CSV file into a DataFrame.

[5]:
```
def load_df(path):
    df = pd.read_csv(path)
    return df
```

Next, I load the subreddit edgelist into a dataframe. All graphs will be made from `reddit_df` or a subset of it.

[6]:
```
reddit_csv = "./data/redditHyperlinks-subredditsOnly.csv"
reddit_df = load_df(reddit_csv)

print(reddit_df.columns)
```

```
Index(['SOURCE_SUBREDDIT', 'TARGET_SUBREDDIT', 'WEIGHT'], dtype='object')
```

The following function is a helper function that plots the nodes of a network via Datashader.

[7]:
```
def nodes_plot(nodes, name=None, canvas=None, cat=None, cmap=["#FF3333"]):
    # Create Datashader Canvas
    canvas = ds.Canvas(**cvsopts) if canvas is None else canvas

    # Create node aggregator
    aggregator = None if cat is None else ds.count_cat(cat)
    agg = canvas.points(nodes, 'x', 'y', aggregator)

    # Return shaded nodes
    return tf.spread(tf.shade(agg, cmap=cmap), px=3, name=name)
```

The following function plots the edges of a network via Datashader

[8]:
```
def edges_plot(edges, name=None, canvas=None):
    # Create Datashader Canvas
    canvas = ds.Canvas(**cvsopts) if canvas is None else canvas

    # Return shaded edges
    return tf.shade(canvas.line(edges, 'x', 'y', agg=ds.count()), name=name)
```

The following function plots the nodes and edges combined

[9]:
```
def graph_plot(nodes, edges, name="", canvas=None, cat=None):
    # Create the Canvas to draw on
    if canvas is None:
        xr = nodes.x.min(), nodes.x.max()
        yr = nodes.y.min(), nodes.y.max()
        canvas = ds.Canvas(x_range=xr, y_range=yr, **cvsopts)
```

```
    # Plot the nodes
    np = nodes_plot(nodes, name + " nodes", canvas, cat)

    # Plot the edges
    ep = edges_plot(edges, name + " edges", canvas)

    # Return both nodes and edges shaded over one another
    return tf.stack(ep, np, how="over", name=name)
```

Now, perform some formatting to allow the nodes and edges to be displayed by the `graph_plot()` function.

[10]:
```
# Create a list of unique subreddits
unique = pd.concat([reddit_df["SOURCE_SUBREDDIT"],␣
 ↪reddit_df["TARGET_SUBREDDIT"]]).unique()

# Create a list of nodes
nodes = pd.DataFrame({'name': unique})

# Create a dictionary of node name to index in nodes
nodes_dict = {name: idx for idx, name in enumerate(nodes['name'])}

# Create list of edges by mapping nodes_dict over reddit_df
edges = reddit_df.copy()
edges['source'] = edges["SOURCE_SUBREDDIT"].map(nodes_dict)
edges['target'] = edges["TARGET_SUBREDDIT"].map(nodes_dict)
edges['weight'] = edges['WEIGHT']
edges.drop(columns=["SOURCE_SUBREDDIT", "TARGET_SUBREDDIT", "WEIGHT"],␣
 ↪inplace=True)

print(nodes.head)
print(edges.head)
```

```
<bound method NDFrame.head of                      name
0                     007
1                   07scape
2             07thexpansion
3          098f6bcd4621d373
4                0_______0
…                      …
67175  radiotransmissions
67176         islandparkny
67177  pogolithearthsound
67178           ifukmydog
67179            zistopia

[67180 rows x 1 columns]>
```

```
<bound method NDFrame.head of         source  target  weight
0             0   55863       1
1             1   34580       2
2             2   53031       1
3             3    3289       1
4             4   49828       1
…           …      …      …
339638    55860   49449       1
339639    55861    6316       1
339640    55861   17220       1
339641    55861   19663       1
339642    55862   49953       1

[339643 rows x 3 columns]>
```

Next I plot the network in a Circular layout and using the ForceAtlas2 algorithm.

[11]:
```
# Run this once to calculate the position of the nodes in both layouts.
# The ForceAtlas2 Algorithm takes a long time for the entire network.
# For example, when testing it took 28 minutes on a 32GB machine.
cd_csv = "./data/circular.csv"
fd_csv = "./data/force_directed.csv"

print("Checking for saved Circular Layout CSV...")
if not path.exists(cd_csv):
    print("File not found, generating layout...")
    %time cd = circular_layout(nodes, uniform=False)
else:
    print(f"Found file at: {cd_csv}")
    cd = pd.read_csv(cd_csv)

print("Checking for saved Force-Directed Layout CSV...")
if not path.exists(fd_csv):
    print("File not found, generating layout...")
    %time fd = forceatlas2_layout(nodes, edges)
else:
    print(f"Found file at: {fd_csv}")
    fd = pd.read_csv(fd_csv)
```

```
Checking for saved Circular Layout CSV…
Found file at: ./data/circular.csv
Checking for saved Force-Directed Layout CSV…
Found file at: ./data/force_directed.csv
```

As this operation can take a long time, save the generated layouts to a CSV file.

[12]:
```
# Save Circular Layout
if not path.exists(cd_csv):
    print(f"Saving file to: {cd_csv}")
```

```
    cd.to_csv(cd_csv)

# Save Force-Directed Layout
if not path.exists(fd_csv):
    print(f"Saving file to: {fd_csv}")
    fd.to_csv(fd_csv)
```

Define some Canvas properties.

[13]:
```
cvsopts = dict(plot_height=1200, plot_width=1200)
```

Create the plots. In this case, create a directly-connected graph for both Circular and Force-Directed as well as Bundled Edge graphs.

[12]:
```
# Plot directly connected edge versions
%time cd_n = graph_plot(cd, directly_connect_edges(cd, edges), "Circular␣
 ↪Layout")
%time fd_n = graph_plot(fd, directly_connect_edges(fd, edges), "Force-Directed␣
 ↪Layout")

# Also plot bundled edge versions
%time cd_b = graph_plot(cd, hammer_bundle(cd, edges, weight='weight'), "Bundled␣
 ↪Circular Layout")
%time fd_b = graph_plot(fd, hammer_bundle(fd, edges, weight='weight'), "Bundled␣
 ↪Force-Directed Layout")
```

```
CPU times: total: 4.7 s
Wall time: 4.74 s
CPU times: total: 3.08 s
Wall time: 3.07 s
CPU times: total: 27min 55s
Wall time: 12min 44s
CPU times: total: 7min 3s
Wall time: 4min 32s
```

[13]:
```
# Display images
tf.Images(cd_n, fd_n, cd_b, fd_b).cols(2)
```

[13]:
```
<datashader.transfer_functions.Images at 0x1cf39dfbd00>
```

As can be seen form the Force-Directed layout, there are many disjoint components which do not connect to the main network of subreddits. These could be considered as very small communities in themselves. There is an evident, large central mass of subreddits, which are all interconnected.

In terms of the Circular layout, there are not any visible or clearly defined communities evident, although it is clear from the thick edges found in the Bundled representation that subreddits tend to connect to one another via key nodes.

As there are many disconnected components, the Louvain algorithm can be sped up by remove these disconnected components before computation so that they do not need to be evaluated.

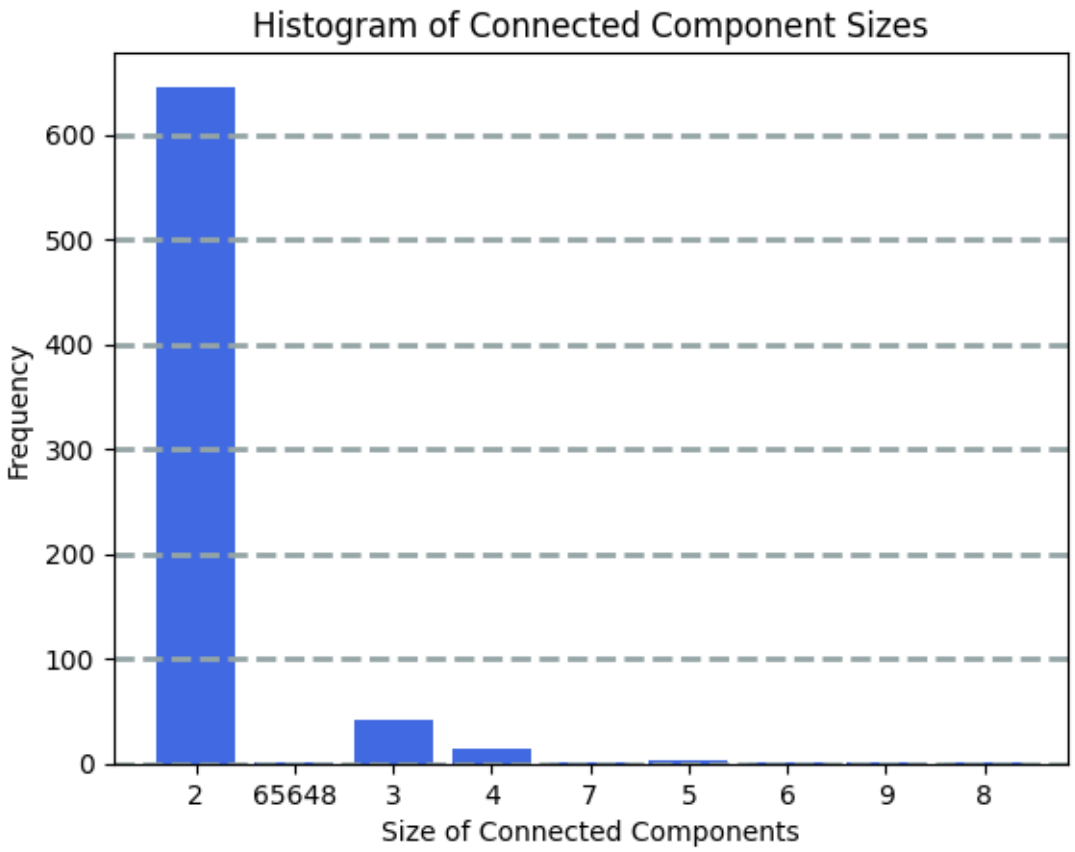
[14]:
```
# Create Weighted Graph
weighted = nx.from_pandas_edgelist(edges, 'source', 'target', ['weight'])
print(f"Number of Nodes: {nx.number_of_nodes(weighted)}")
```

```
Number of Nodes: 67180
```

To evaluate the amount of trimming necessary, create a histogram showing the size of all connected components in the dataset.

[15]:
```
# Get all connected components
%time connected_components = list(nx.connected_components(weighted))
print(f"There are {len(connected_components)} connected components.")

# Get sizes of connected components
component_sizes = [len(c) for c in connected_components]
size_counts = Counter(component_sizes)

size_labels = size_counts.keys()

# Plot histogram
plt.xticks(range(len(size_counts.values())), size_labels)
plt.xlabel('Size of Connected Components')
plt.ylabel('Frequency')
plt.title('Histogram of Connected Component Sizes')
plt.bar(range(len(size_counts.values())), size_counts.values(),␣
 ↪color='royalblue')
plt.grid(color='#95a5a6', linestyle='--', linewidth=2, axis='y')
plt.show()

# Print list of actual numbers
for key, value in size_counts.items():
    if value == 1:
        print(f"There is {value} component that contains {key} nodes.")
    else:
        print(f"There are {value} components that contain {key} nodes.")
```

```
CPU times: total: 93.8 ms
Wall time: 90.3 ms
There are 712 connected components.
```




```
There are 646 components that contain 2 nodes.
There is 1 component that contains 65648 nodes.
There are 42 components that contain 3 nodes.
There are 14 components that contain 4 nodes.
There are 2 components that contain 7 nodes.
There are 3 components that contain 5 nodes.
There are 2 components that contain 6 nodes.
There is 1 component that contains 9 nodes.
There is 1 component that contains 8 nodes.
```

As can be seen from the histogram, there are 646 components of only 2 nodes, which can be immediately discarded. Going forwards, I will use the single connected component containing 65648 nodes, as this contains enough nodes to apply community detection to.

[16]:
```
# Get largest connected component
largest = max(connected_components, key=len)
%time largest = weighted.subgraph(largest)
print(f"Number of Nodes: {nx.number_of_nodes(largest)}")
```

```
CPU times: total: 0 ns
Wall time: 6.94 ms
```

```
Number of Nodes: 65648
```

Now plot the Circular and Force-Directed layouts again for the new, fully-connected network.

[17]:
```
# Convert back into edgelist dataframe
%time connected_edges = nx.to_pandas_edgelist(largest)
print(f"Number of Edges: {len(connected_edges)}")
print(connected_edges.columns)
```

```
CPU times: total: 3.83 s
Wall time: 3.83 s
Number of Edges: 308839
Index(['source', 'target', 'weight'], dtype='object')
```

[18]:
```
# Create a new node list for the large component
in_largest = nodes.index.isin(set(connected_edges['source']) |␣
 ↪set(connected_edges['target']))
connected_nodes = nodes[in_largest]

print(connected_nodes.head)
print(connected_edges.head)
```

```
<bound method NDFrame.head of                       name
1                  07scape
2            07thexpansion
3          098f6bcd4621d373
4                0_______0
5                  0magick
…                      …
67175  radiotransmissions
67176         islandparkny
67177  pogolithearthsound
67178           ifukmydog
67179            zistopia

[65648 rows x 1 columns]>
<bound method NDFrame.head of         source  target  weight
0            1   34580       2
1            1     117       3
2        34580     117       4
3        34580   47740      10
4            2   53031       1
…        …     …     …
308834   55729   67169       1
308835   55729   67170       1
308836   55729   67171       1
308837   55796   55850       1
308838   55798   55799       1
```

```
[308839 rows x 3 columns]>
```

Calculate the positions for the single, connected component

[19]:
```
# Calculate positions for the Circular and Force-Directed Layouts
# The ForceAtlas2 Algorithm takes a long time for the entire network.
# For example, when testing it took 31 minutes on a 32GB machine.
connected_cd_csv = "./data/connected_cd.csv"
connected_fd_csv = "./data/connected_fd.csv"

print("Checking for saved Connected Circular layout CSV...")
if not path.exists(connected_cd_csv):
    print("File not found, generating layout...")
    %time connected_cd = circular_layout(connected_nodes, uniform=False)
else:
    print(f"File found at: {connected_cd_csv}")
    connected_cd = pd.read_csv(connected_cd_csv)

print("Checking for saved Connected Force-Directed layout CSV...")
if not path.exists(connected_fd_csv):
    print("File not found, generating layout...")
    %time connected_fd = forceatlas2_layout(connected_nodes, edges)
else:
    print(f"File found at: {connected_fd_csv}")
    connected_fd = pd.read_csv(connected_fd_csv)
```

```
Checking for saved Connected Circular layout CSV…
File found at: ./data/connected_cd.csv
Checking for saved Connected Force-Directed layout CSV…
File found at: ./data/connected_fd.csv
```

Again, the above operation can take a long time, so save the DataFrame to CSV for later use.

[20]:
```
# Save Circular Layout
if not path.exists(connected_cd_csv):
    print(f"Saving file to: {connected_cd_csv}")
    connected_cd.to_csv(connected_cd_csv)

# Save Force-Directed Layout
if not path.exists(connected_fd_csv):
    print(f"Saving file to: {connected_fd_csv}")
    connected_fd.to_csv(connected_fd_csv)
```

[20]:
```
# Plot directly connected edge versions
%time cd_n = graph_plot(connected_cd, directly_connect_edges(connected_cd,␣
 ↪connected_edges), "Connected Circular Layout")
%time fd_n = graph_plot(connected_fd, directly_connect_edges(connected_fd,␣
 ↪connected_edges), "Connected Force-Directed Layout")
```

```
# Also plot bundled edge versions
%time cd_b = graph_plot(connected_cd, hammer_bundle(connected_cd,␣
 ↪connected_edges, weight='weight'), "Connected Bundled Circular Layout")
%time fd_b = graph_plot(connected_fd, hammer_bundle(connected_fd,␣
 ↪connected_edges, weight='weight'), "Connected Bundled Force-Directed Layout")
```

```
CPU times: total: 2.33 s
Wall time: 2.33 s
CPU times: total: 1.59 s
Wall time: 1.59 s
CPU times: total: 26min 35s
Wall time: 11min 36s
CPU times: total: 8min 17s
Wall time: 5min 33s
```

[21]:
```
# Display images
tf.Images(cd_n, fd_n, cd_b, fd_b).cols(2)
```

[21]:
```
<datashader.transfer_functions.Images at 0x1cf415b4e50>
```

Now that there is a single, connected network, community detection can be carried out. First, the Louvain algorithm:

[21]:
```
# Make the new network
connected = nx.from_pandas_edgelist(connected_edges)
print(f"Number of Nodes: {nx.number_of_nodes(connected)}")
```

```
Number of Nodes: 65648
```

[22]:
```
# Make dictionary of community labels
%time louvain_dict = community.best_partition(connected)

# Unfold dict into a DataFrame
%time louvain_labels = pd.DataFrame(list(louvain_dict.items()),␣
 ↪columns=['node', 'community'])

print(f"There are {louvain_labels['community'].nunique()} communities")

print(louvain_labels.head)
```

```
CPU times: total: 22.1 s
Wall time: 22.1 s
CPU times: total: 31.2 ms
Wall time: 27.3 ms
There are 75 communities
<bound method NDFrame.head of         node  community
0          1          0
1      34580          0
2        117          0
```

```
3      47740          0
4          2          0
…      …        …
65643  67167          0
65644  67169         15
65645  67170         15
65646  67171         15
65647  55798          7

[65648 rows x 2 columns]>
```

Following function gets a set of colours equally spaced across a given colormap, for the display of communities.

[23]:
```python
def get_colours(n, cmap):
    c_map = plt.colormaps[str(cmap)]

    arr = np.linspace(0, 1, n)
    color_list = list()

    for c in arr:
        rgba = c_map(c)
        clr = mcolors.rgb2hex(rgba)
        color_list.append(str(clr))

    return color_list
```

Following function plots categorised nodes with the previous calculated Circular and Force-Directed node positions.

[24]:
```python
def plot_categories(nodes, edges, labels, name="", cat=None):
    n_comms = labels['community'].nunique()

    community_labels = np.sort(labels['community'].unique())

    # Get colours for all communities
    colours = get_colours(n_comms, "hsv")

    # Create canvas
    xr = nodes.x.min(), nodes.x.max()
    yr = nodes.y.min(), nodes.y.max()
    canvas = ds.Canvas(x_range=xr, y_range=yr, **cvsopts)

    node_plots = []

    for community in community_labels:
        # DataFrame of nodes in current community
        community_list = labels.loc[labels['community'] == community]
```

```
        # Plot subset of nodes with specific colour
        sub_nodes = nodes.loc[nodes.index.isin(community_list['node'])]

        node_plot = nodes_plot(sub_nodes, f"Community {community}", canvas,␣
 ↪cat=cat, cmap=colours[community])
        node_plots.append(node_plot)

    edge_plot = edges_plot(edges, name + " edges", canvas)

    return tf.stack(edge_plot, *node_plots, how="over", name=name)
```

[106]:
```
# Plot the communities using the same layout as found previously
%time cat_cd = plot_categories(connected_cd,␣
 ↪directly_connect_edges(connected_cd, connected_edges), louvain_labels,␣
 ↪name="Circular Layout Categorised")
%time cat_fd = plot_categories(connected_fd,␣
 ↪directly_connect_edges(connected_fd, connected_edges), louvain_labels,␣
 ↪name="Force-Directed Layout Categorised")

tf.Images(cat_cd, cat_fd).cols(2)
```

```
CPU times: total: 4.78 s
Wall time: 4.79 s
CPU times: total: 4.02 s
Wall time: 4.02 s
```

[106]:
```
<datashader.transfer_functions.Images at 0x168e48784f0>
```

These visualisations do not show evident communities as there are too many nodes, too closely clustered together. Next, create a grid of images showing each community by itself.

[25]:
```
def plot_single_community(nodes, edges, labels, n, name="", cat=None):
    n_comms = labels['community'].nunique()

    community_labels = np.sort(labels['community'].unique())

    # Get colours for all communities
    colours = get_colours(n_comms, "hsv")
    col = colours[n]

    # Create canvas
    xr = nodes.x.min(), nodes.x.max()
    yr = nodes.y.min(), nodes.y.max()
    canvas = ds.Canvas(x_range=xr, y_range=yr, **cvsopts)

    # DataFrame of nodes in current community
    community_list = labels.loc[labels['community'] == n]
```

```
    # Plot subset of nodes with specific colour
    sub_nodes = nodes.loc[nodes.index.isin(community_list['node'])]

    sub_node_names = set(sub_nodes['name'])

    node_plot = nodes_plot(sub_nodes, f"Community {n}", canvas, cat=cat,␣
 ↪cmap=col)

    edges_pos = directly_connect_edges(sub_nodes, edges)

    edge_plot = edges_plot(edges_pos, name + " edges_pos", canvas)

    return tf.stack(edge_plot, node_plot, how="over", name=name)
```

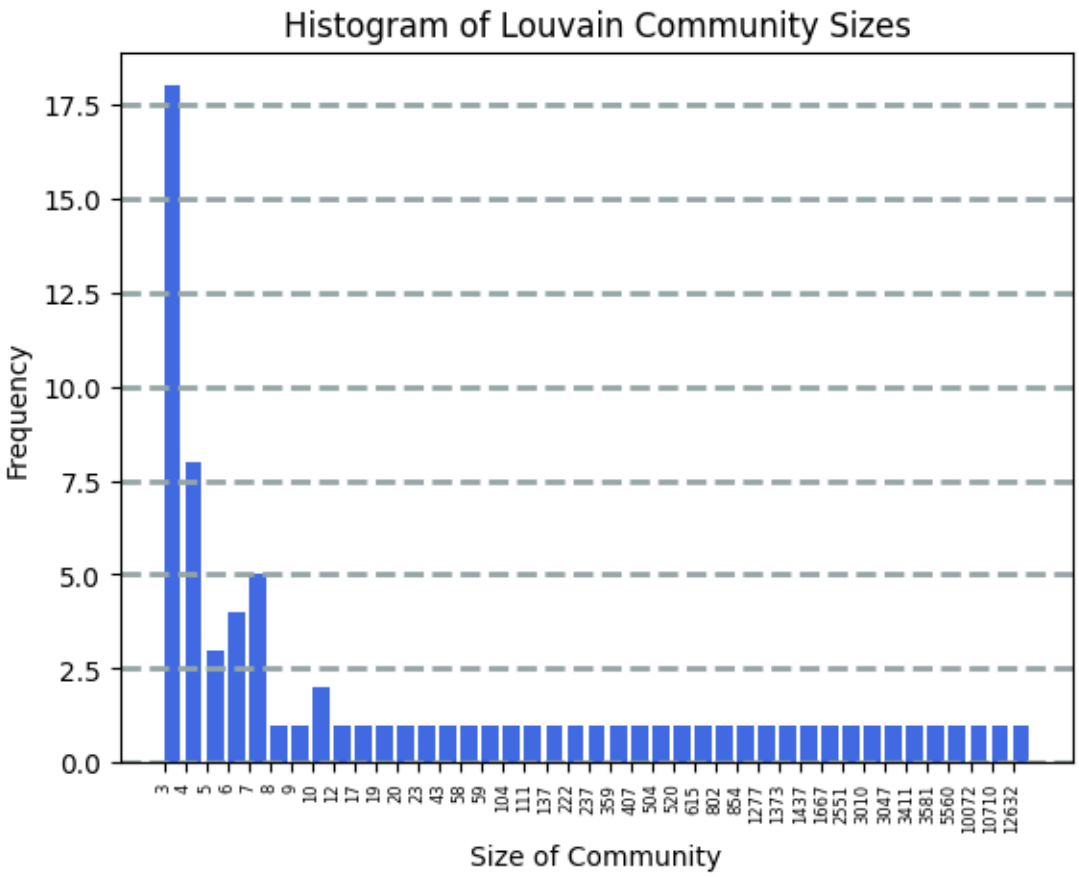
[29]:
```
imgs = []
for n in range(0, louvain_labels['community'].nunique()):
    single_cd = plot_single_community(connected_cd, connected_edges,␣
 ↪louvain_labels, n, name=f"Circular, Community {n}")
    single_fd = plot_single_community(connected_fd, connected_edges,␣
 ↪louvain_labels, n, name=f"Force-Directed, Community {n}")

    imgs.append(single_cd)
    imgs.append(single_fd)

tf.Images(*imgs).cols(2)
```

[29]:
```
<datashader.transfer_functions.Images at 0x1d034a8a850>
```

[26]:
```
# Plot histogram of Louvain Communities
louvain_hist = louvain_labels.groupby('community').size().
 ↪reset_index(name='number')
louvain_hist = louvain_hist.groupby('number').size().
 ↪reset_index(name='frequency').rename(columns={'number': 'size'})

print(louvain_hist.head)

# Plot histogram of Louvain Communities
plt.xticks(range(len(louvain_hist['frequency'])), louvain_hist['size'],␣
 ↪rotation='vertical', fontsize=6)

# Add axis labels and a title
plt.xlabel('Size of Community')
plt.ylabel('Frequency')
plt.title('Histogram of Louvain Community Sizes')
plt.bar(range(len(louvain_hist['frequency'])), louvain_hist['frequency'],␣
 ↪color='royalblue', align='edge', width=0.8)
```

```
plt.grid(color='#95a5a6', linestyle='--', linewidth=2, axis='y')
plt.show()
```

```
<bound method NDFrame.head of      size  frequency
0       3         18
1       4          8
2       5          3
3       6          4
4       7          5
5       8          1
6       9          1
7      10          2
8      12          1
9      17          1
10     19          1
11     20          1
12     23          1
13     43          1
14     58          1
15     59          1
16    104          1
17    111          1
18    137          1
19    222          1
20    237          1
21    359          1
22    407          1
23    504          1
24    520          1
25    615          1
26    802          1
27    854          1
28   1277          1
29   1373          1
30   1437          1
31   1667          1
32   2551          1
33   3010          1
34   3047          1
35   3411          1
36   3581          1
37   5560          1
38  10072          1
39  10710          1
40  12632          1>
```



Next, perform Asynchronous Label Propagation algorithm

[27]:
```
%time lpa = comm.asyn_lpa_communities(connected, weight='weight')
```

```
CPU times: total: 0 ns
Wall time: 0 ns
```

[28]:
```
%time lpa_list = list(lpa)
```

```
CPU times: total: 5.39 s
Wall time: 5.4 s
```

[29]:
```
nodes_to_add = []
communities_to_add = []

for i, sublist in enumerate(lpa_list):
    for node in sublist:
        nodes_to_add.append(node)
        communities_to_add.append(i)
```

```
to_add = {'node': nodes_to_add, 'community': communities_to_add}
lpa_labels = pd.DataFrame.from_dict(to_add)

print(lpa_labels.head)
```

```
<bound method NDFrame.head of         node  community
0          1          0
1          2          0
2          3          0
3          4          0
4          5          0
…        …        …
65643  46162       1364
65644  47216       1365
65645  65781       1365
65646  65784       1366
65647  47258       1366

[65648 rows x 2 columns]>
```

[113]:
```
# Plot the communities using the same layout as found previously
%time cat_cd = plot_categories(connected_cd,␣
 ↪directly_connect_edges(connected_cd, connected_edges), lpa_labels,␣
 ↪name="Circular Layout Categorised")
%time cat_fd = plot_categories(connected_fd,␣
 ↪directly_connect_edges(connected_fd, connected_edges), lpa_labels,␣
 ↪name="Force-Directed Layout Categorised")

tf.Images(cat_cd, cat_fd).cols(2)
```

```
CPU times: total: 1min 9s
Wall time: 1min 9s
CPU times: total: 1min
Wall time: 1min
```

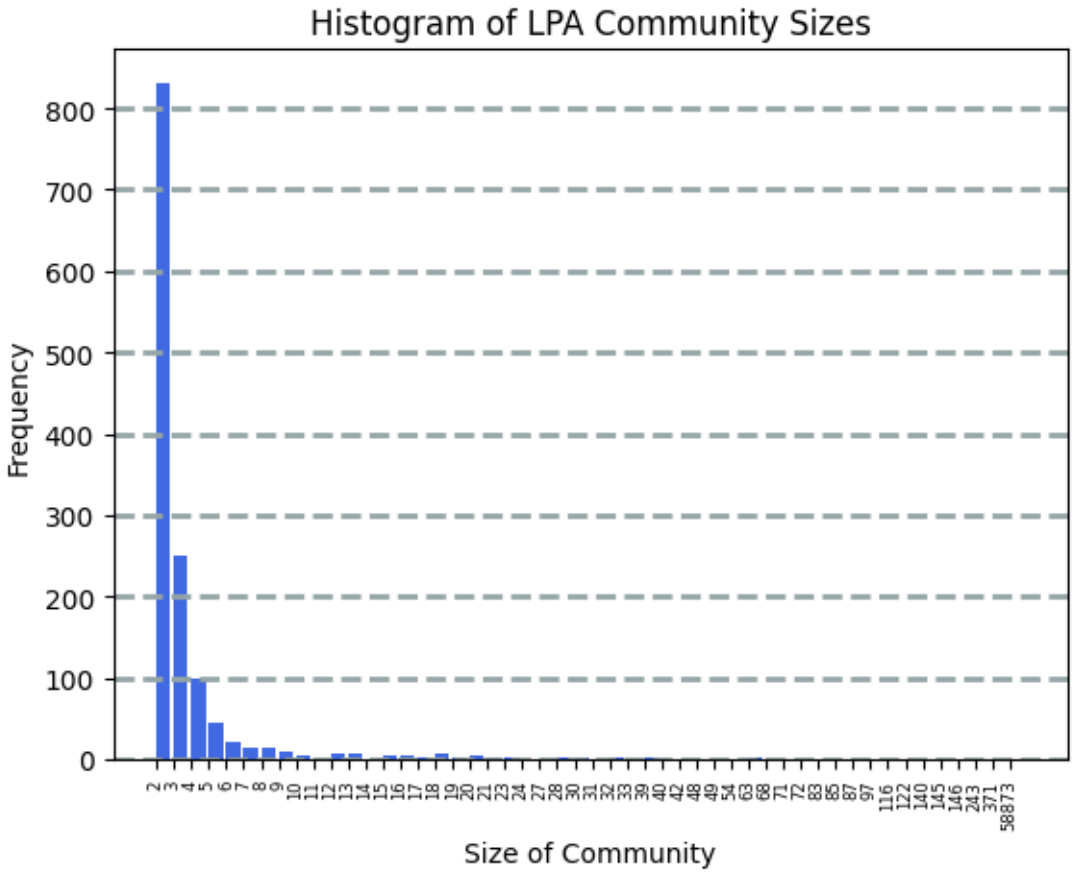
[113]:
```
<datashader.transfer_functions.Images at 0x1688e70ab80>
```

[30]:
```
hist = lpa_labels.groupby('community').size().reset_index(name='number')
hist = hist.groupby('number').size().reset_index(name='frequency').
 ↪rename(columns={'number': 'size'})

print(hist.head)

# Plot a histogram of the 'number' column
plt.xticks(range(len(hist['frequency'])), hist['size'], rotation='vertical',␣
 ↪fontsize=6)

# Add axis labels and a title
```

```
plt.xlabel('Size of Community')
plt.ylabel('Frequency')
plt.title('Histogram of LPA Community Sizes')
plt.bar(range(len(hist['frequency'])), hist['frequency'], color='royalblue',␣
 ↪align='edge', width=0.8)
plt.grid(color='#95a5a6', linestyle='--', linewidth=2, axis='y')
plt.show()
```

```
<bound method NDFrame.head of      size  frequency
0       2        831
1       3        251
2       4         99
3       5         45
4       6         22
5       7         14
6       8         15
7       9          9
8      10          5
9      11          1
10     12          7
11     13          6
12     14          1
13     15          4
14     16          4
15     17          2
16     18          6
17     19          2
18     20          4
19     21          3
20     23          2
21     24          1
22     27          1
23     28          2
24     30          2
25     31          1
26     32          2
27     33          1
28     39          2
29     40          1
30     42          1
31     48          1
32     49          1
33     54          1
34     63          2
35     68          1
36     71          1
37     72          1
38     83          1
```

```
39      85           1
40      87           1
41      97           1
42     116           1
43     122           1
44     140           1
45     145           1
46     146           1
47     243           1
48     371           1
49   58873           1>
```



Calculate TPR, Conductance and Modularity of Louvain Labels and LPA Labels.

[59]:
```
# Convert Louvain Labels into a NodeClustering object, so that I can
# perform evaluation on them
louvain_list = [[] for _ in range(max(louvain_dict.values()) + 1)]

for node, community in louvain_dict.items():
    louvain_list[community].append(node)
```

```
louvain_coms = NodeClustering(louvain_list, graph=connected,␣
 ↪method_name='Louvain')
```

[60]:
```
# Calculate Conductance, TPR and Modularity:
louvain_conductance = conductance(connected, louvain_coms, summary=True)
louvain_tpr = triangle_participation_ratio(connected, louvain_coms,␣
 ↪summary=True)
louvain_modularity = newman_girvan_modularity(connected, louvain_coms,␣
 ↪summary=True)

print(f"Conductance: {louvain_conductance}")
print(f"TPR: {louvain_tpr}")
print(f"Modularity: {louvain_modularity}")
```

```
Conductance: FitnessResult(min=0.0027247956403269754, max=0.6515609264853978,
score=0.23753507740180022, std=0.13220832166822985)
TPR: FitnessResult(min=0.0, max=1.0, score=0.24036557171981135,
std=0.2877272691911959)
Modularity: FitnessResult(min=None, max=None, score=0.41738497141682823,
std=None)
```

[61]:
```
# Convert the LPA labels into a NodeClustering object, so that I can
# perform evaluation on them
lpa_coms = NodeClustering(lpa_list, graph=connected, method_name='LPA')
```

[62]:
```
# Calculate Conductance, TPR and Modularity:
lpa_conductance = conductance(connected, lpa_coms, summary=True)
lpa_tpr = triangle_participation_ratio(connected, lpa_coms, summary=True)
lpa_modularity = newman_girvan_modularity(connected, lpa_coms, summary=True)

print(f"Conductance: {lpa_conductance}")
print(f"TPR: {lpa_tpr}")
print(f"Modularity: {lpa_modularity}")
```

```
Conductance: FitnessResult(min=0.008459009567176759, max=0.7142857142857143,
score=0.36287489337285733, std=0.10288671520150713)
TPR: FitnessResult(min=0.0, max=1.0, score=0.03816279882849793,
std=0.15726532973048263)
Modularity: FitnessResult(min=None, max=None, score=0.05204184799923872,
std=None)
```