A Movie Network of Shared Tropes

February 2, 2023

1 Math 1010 Final Project

Isaac, Tianyu, Tim

[1]:
```
import pandas as pd
import networkx as nx
import numpy as np
import os
import matplotlib.pyplot as plt
from networkx.algorithms import bipartite
import copy
import matplotlib.cm as cmx

folder = ""
file = "m2mel_pandas_filtered_smol"
ext = ".csv"

m2m_pd = pd.read_csv( folder+file+ext )
m2m_nx = nx.from_pandas_edgelist(m2m_pd, source = "n1", target = "n2",␣
 ↪edge_attr = True, create_using = nx.Graph)
```

[2]:
```
#Finding the largest component
m2m_comp_sets = list(nx.connected_components(m2m_nx))
m2m_comp_sets.sort(reverse = True,key = len)
#Making a new network
m2m_nx_big = copy.deepcopy(m2m_nx)
#Removing nodes that are not in the largest component.
m2m_nx_big.remove_nodes_from([n for n in list(m2m_nx_big.nodes()) if n not in␣
 ↪m2m_comp_sets[0]])
#Weights
weights_bign = nx.get_edge_attributes(m2m_nx_big,'weight').values()
#Print the information on nodes and edges
print("The graph has",m2m_nx_big.number_of_nodes(),"nodes.")
print("The graph has",m2m_nx_big.number_of_edges(),"edges.")
```

```
The graph has 122 nodes.
The graph has 1903 edges.
```

- Learn about the network dataset you've been assigned. Describe it for us.

- what are the nodes?
    - How many nodes are there?
- what are the edges?
    - How many edges are there?
- Is it weighted or unweighted?
    - If weighted, what do the weights represent/mean?
- directed or undirected?
- is it connected or disconnected? if disconnected, is there a giant component?
- Is there anything interesting or weird about it?
- What's the context of this network?
- Do you have access to any metadata about the nodes?

1.0.1 Network information

This is a network showing how similar different movies are as measured by common tropes. Two movies have an edge between them if they share at least 25 tropes, and the weight of the edge is based on the number of tropes they share between each other. In other words, the network is undirected and weighted by construction.

It is constructed in the following manner. First, we took the top 50 movies by US domestic box office from 2015-2018 using IMDB. For each movie, we looked at their tvtropes page. Tvtropes is a community that lists the tropes for each movie. The site defines a trope as "a storytelling device or convention...[and] the means by which a story is told by anyone who has a story to tell. One can think of them as repeated themes or motifs that are seen throughout stories. Popular tropes include Checkov's gun, call-back, and running gags.

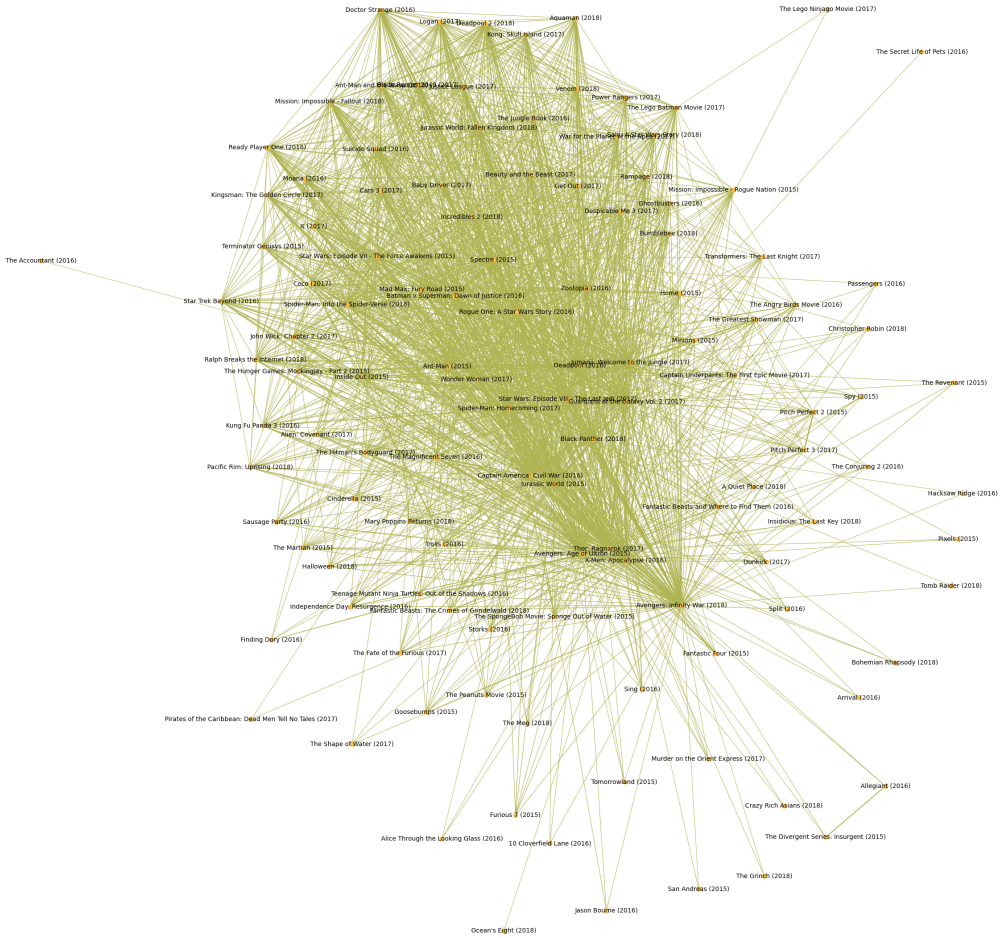
Using Rvest, I scraped the tropes pages for every movie listed on the site, and merged it with a dataset with imdb data. Movies that couldn't be merged or didn't have a tv tropes page was dropped. Some tropes are purposefully averted at some point in the movie, and those are dropped from the analysis. I first construct a bipartite network between movies and tropes, and then project the network onto the set of movie nodes. The weight between two movies is thus the number of tropes they share with each other. I filter my edgelist to edges that weigh at least 25. Finally, I took the largest component of the network and deleted the smaller components (which are often movies from the same series or remakes of the same movie). The original attempt at constructing the network that used movies from every year imdb has data on yielded 52 components, and I thought focusing on the largest component would be more interesting.

In the end, the process yielded a connected network with 122 nodes and 1903 edges. One can easily access metadata on the nodes by googling the movie, and all data on edges are calculated from publicly availiable data on tvtropes.org. The most interesting about the network is how densely connected all of the movies are. This makes sense when we think about how the network is constructed: the way the projection works is that, for any set of 25 tropes, if a set of movies have all 25 of them then there are going to be edges between every single movie to each other. In other words, those set of movies will have a clustering coefficient of one.

1.0.2 Adjacency matrix and plot

[3]:
```
#Adjacency matrix
adj_matrix = nx.adjacency_matrix(m2m_nx_big)
```

[4]:
```
#Plot
plt.figure(figsize = (25,25))
nx.draw(m2m_nx_big, pos = nx.kamada_kawai_layout(m2m_nx_big), node_color =␣
 ↪'#ffbd4a',
        width = [(i/50)**0.5 for i in list(weights_bign)], font_size = 10,␣
 ↪node_size = 50,
        edge_color = "#afb352",
        with_labels = True)
plt.savefig("tvtropes_net.png", format="PNG")
```



[5]:
```
#Edge weights distribution.
m2m_pd["weight"].describe()
```

[5]:
```
count    1905.000000
mean       36.900787
std        13.868876
min        25.000000
25%        28.000000
50%        33.000000
75%        41.000000
max       203.000000
Name: weight, dtype: float64
```

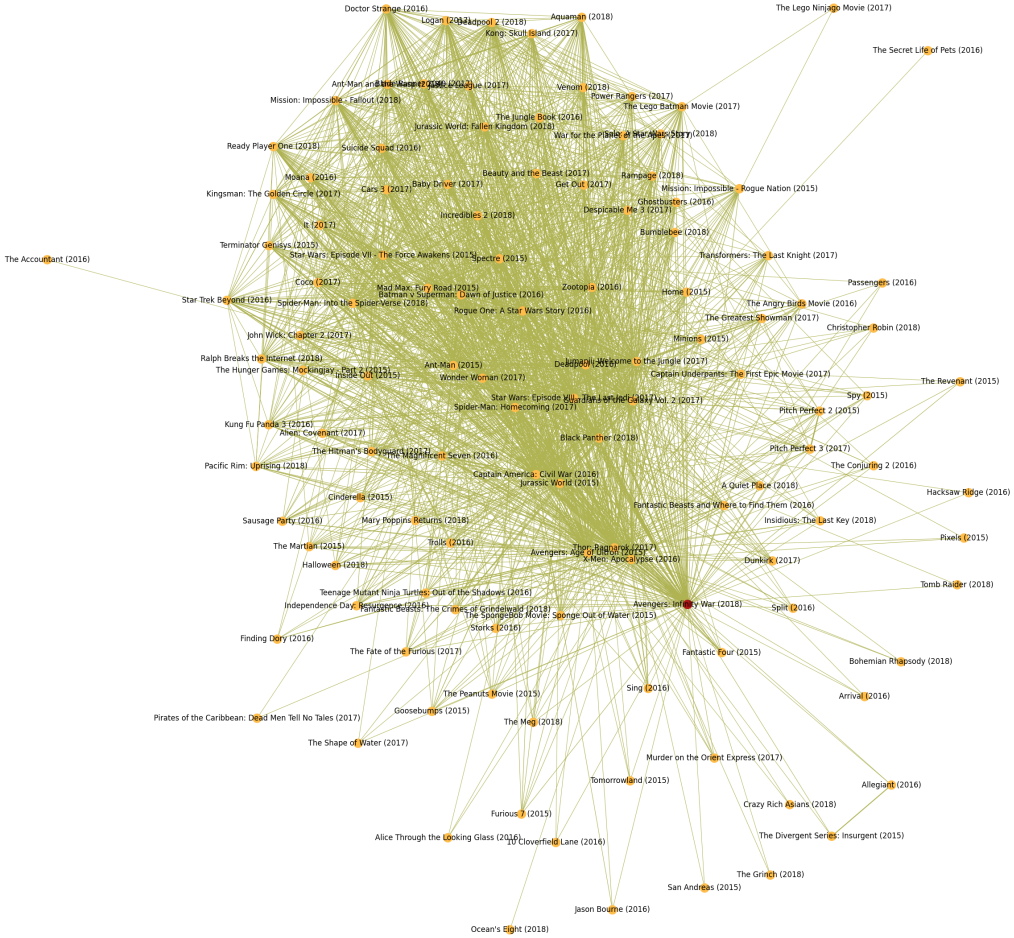
1.0.3 Centrality

- Find the most central node(s) in your network.
  - The most central node is Advengers: Infinity War (2018), with the highest degree, closeness, eigenvector, and betweenness centrality.
  - Do the centrality metrics identify different node(s) as the "most central"?
    - Each centrality metric identify the same node, Avengers: Infinity War (2018), as the most central.
  - Given the context of your dataset, interpret which centrality metric you think is most informative. Explain.
    - Degree centrality is the most informative in our case as it measures which movie incorporates the most tropes from other movies. The most central movie is "cliche" as it uses tropes that appear in many other movies.

[6]:
```
#Centrality Metrics
degreeC = nx.degree_centrality(m2m_nx_big)
closenessC = nx.closeness_centrality(m2m_nx_big)
eigenvectorC = nx.eigenvector_centrality(m2m_nx_big)
betweennessC = nx.betweenness_centrality(m2m_nx_big)

print('The movie with the highest degree centrality is', max(degreeC,␣
 ↪key=lambda key: degreeC[key]))
print('The movie with the highest closeness centrality is', max(closenessC,␣
 ↪key=lambda key: closenessC[key]))
print('The movie with the highest eigenvector centrality is', max(eigenvectorC,␣
 ↪key=lambda key: eigenvectorC[key]))
print('The movie with the highest betweenness centrality is', max(betweennessC,␣
 ↪key=lambda key: betweennessC[key]))

##Plot the figure with the most central node highlighted.
plt.figure(figsize = (25,25))
kawai_pos = nx.kamada_kawai_layout(m2m_nx_big)
```

```
color = ['#ae1717' if node == 'Avengers: Infinity War (2018)' else '#ffbd4a'␣
 ↪for node in m2m_nx_big.nodes()]
nx.draw(
    m2m_nx_big,
    pos = kawai_pos,
    node_color = color,
    width = [(i/50)**0.5 for i in list(weights_bign)],
    font_size = 12,
    node_size = 200,
    edge_color = "#afb352",
    with_labels = True
       )

plt.savefig("KamadaKawai_smol.png", format="PNG")
```

```
The movie with the highest degree centrality is Avengers: Infinity War (2018)
The movie with the highest closeness centrality is Avengers: Infinity War (2018)
The movie with the highest eigenvector centrality is Avengers: Infinity War
(2018)
The movie with the highest betweenness centrality is Avengers: Infinity War
(2018)
```

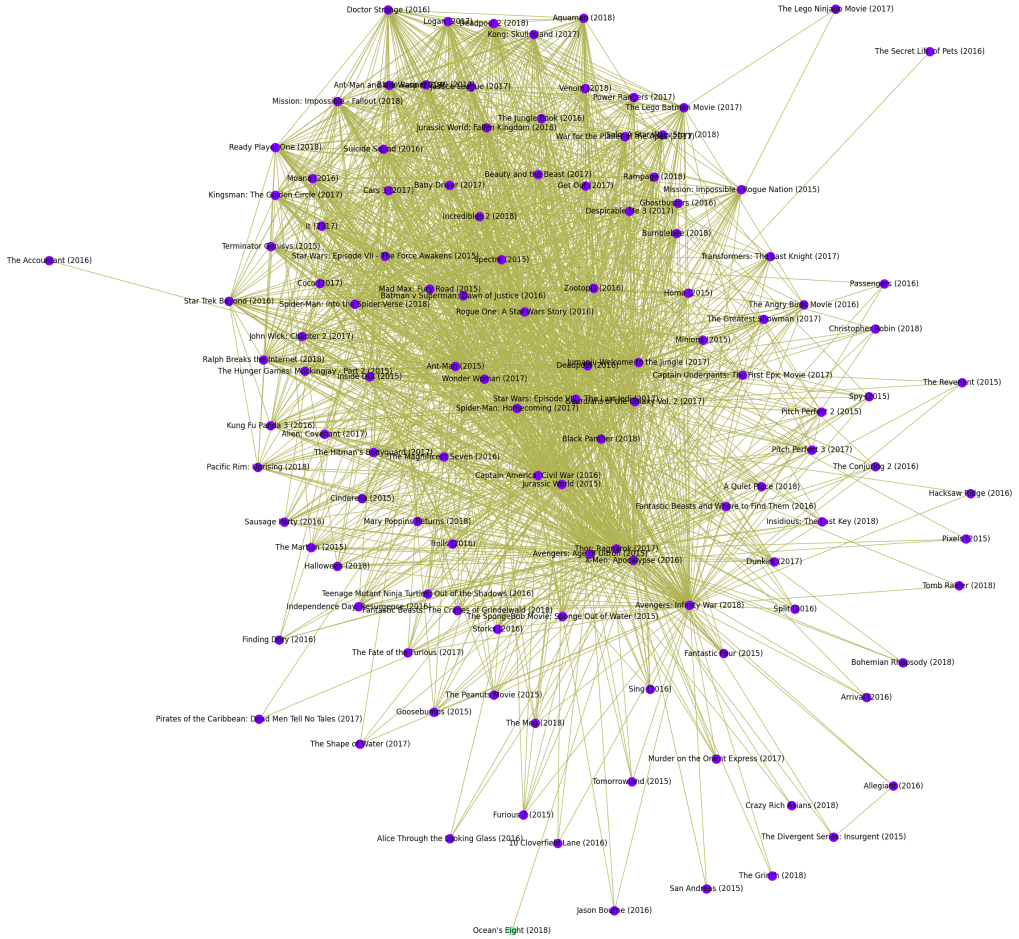


1.0.4 Community

- Find and plot the communities in your network
  - Do the identified communities make sense given the context of your data?
    - Ocean's Eight (2018) is itself a community, but every other node in this network belongs to a large community. Ocean's Eight has the lowest clustering coefficient and is only connected to X-Men: Apocalypse (2016). We suspect that there should exist more communities in this network as some other nodes (The Accountant (2016)) also have a clustering coefficient of 0.

[7]:
```
#Community
def Plot_Comm(Network, C, position = None):
    cmap = cmx.get_cmap(name='rainbow')
```

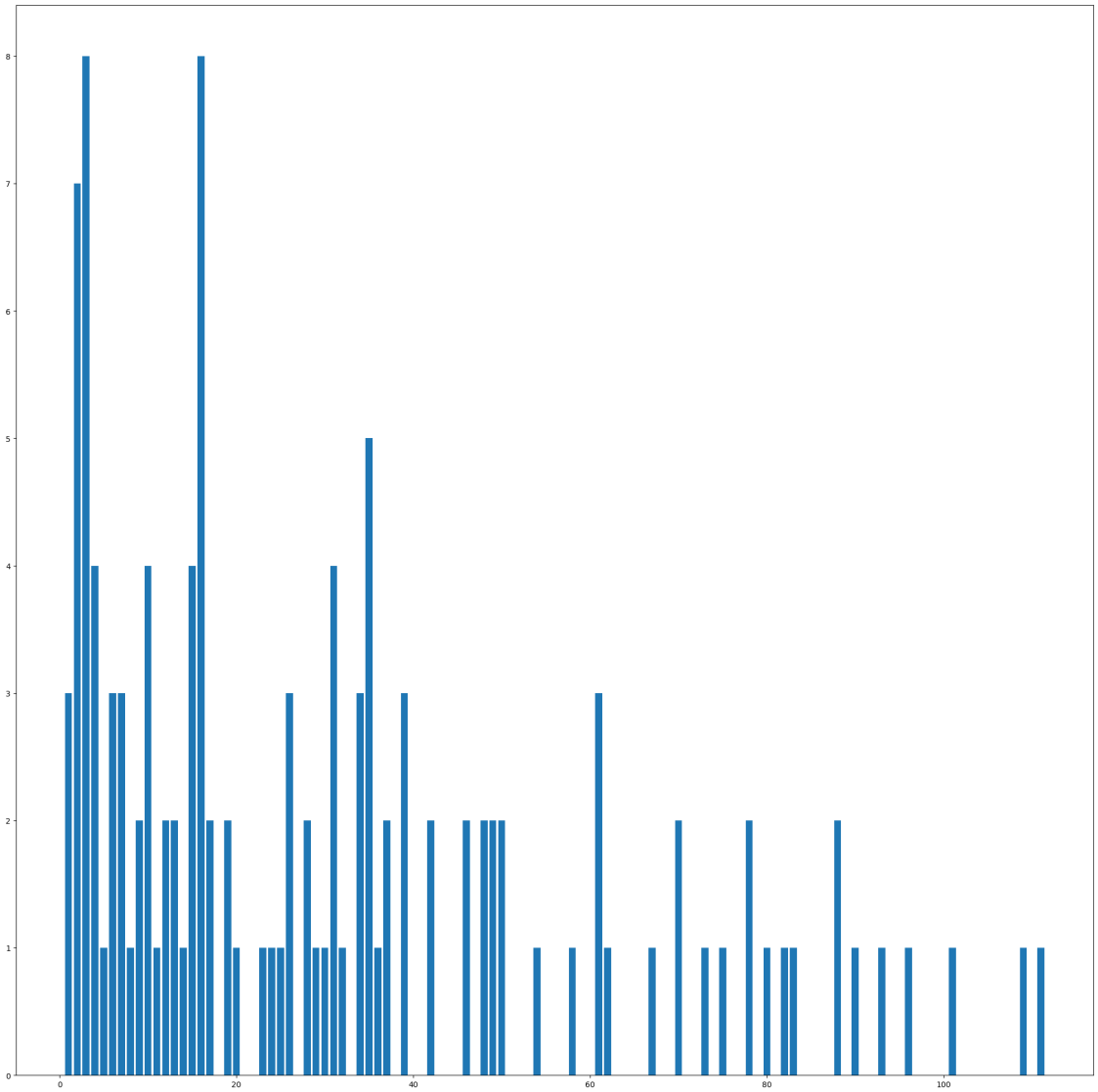
```
    N = len(Network.nodes())
    K = len(C)
    color_map = ['k']*N
    for i in range(K):
        for j in range(len(C[i])):
            color_map[ C[i][j] ] = cmap(i/K)
    if position is None:
        pos = nx.kamada_kawai_layout(Network,iterations=20)
    else:
        pos = position
    figure = plt.figure(figsize = (25,25))
    nx.draw(Network, pos, node_color=color_map, node_size=200, font_size = 12,␣
 ↪edge_color='#afb352', with_labels=True)
    plt.savefig("Community.png", format="PNG")
    plt.show()
    return

##Rename the nodes to be integers.
names = list(m2m_nx_big.nodes())
name_dictionary = { names[i]:i for i in range(0, len(names) ) }
m2m_nx_big_renamed = nx.relabel_nodes(m2m_nx_big, name_dictionary)
##Find communities using the Girvan-Newman method.
comm = nx.algorithms.community.girvan_newman(m2m_nx_big_renamed)
m2m_communities = tuple(sorted(c) for c in next(comm))

kawai_pos = nx.kamada_kawai_layout(m2m_nx_big)
Plot_Comm(m2m_nx_big, m2m_communities, kawai_pos)
```



1.0.5 Degree distribution

[8]:
```
#Degree Distribution
## Each "bin" tells us the number of degrees and the height of the bar tells us␣
 ↪how many nodes have the same degree
plt.figure(figsize = (25,25))
degree_sequence = sorted((d for n, d in m2m_nx_big.degree()), reverse=True)
plt.bar(*np.unique(degree_sequence, return_counts=True))
```

[8]:
```
<BarContainer object of 57 artists>
```



1.0.6 Average shortest path

- The average shortest path in this network is 1.76.

[9]:
```
#Average Shortest Path
print(nx.average_shortest_path_length(m2m_nx_big))
```

```
1.7618208914781195
```

1.0.7 Average clustering coefficient

- The average clustering coefficient in this network is 0.84.
  - The node with the lowest clustering coefficient is Ocean's Eight (2018).
  - The node with the highest clustering coefficient is Fantastic Four (2015).

[10]:
```
#Average Clustering Coefficient

clustering = nx.algorithms.clustering(m2m_nx_big)
print('The node with lowest clustering coefficient is', min(clustering,␣
 ↪key=lambda key: clustering[key]))
print('The node with the highest clustering coefficient is', max(clustering,␣
 ↪key=lambda key: clustering[key]))

average_cc = nx.algorithms.average_clustering(m2m_nx_big)
print('The average clustering coefficient is ', average_cc)
```

```
The node with lowest clustering coefficient is Ocean's Eight (2018)
The node with the highest clustering coefficient is Fantastic Four (2015)
The average clustering coefficient is  0.8419818484205546
```

1.0.8 Conclude

- Do you think about this dataset differently than you did before?
  - We learned that movies are a lot more similar to each other than we previously thought. The clustering coefficient is quite high (0.84) and the average shortest path is 1.76 which shows that the graph is very connected. There are a lot of tropes the movies in our dataset share, the median edge weight of shared tropes is about 33 (by construction we deleted edges with weight less than 25).
- Did you learn anything about the discipline/domain the network lies in?
  - Popular movies are connected to each other by the tropes they share. The density in this network shows us that editors on tvtropes.org are very good at finding these common patterns of storytelling in movies with very different contexts. We see that movies about very different stories (e.g., Cars 3 and Jumanji) still use a similar set of tropes to entertain their audience.
- What's something you wish you could learn about the network that you might not have a tool for?
  - The Girvan-Newman method seems to fail to identify communities in a network with a high clustering coefficient. I wonder if there is a community-finding algorithm that takes into account the weights on the edges and can function in a densely-connected network.
- Is there any metadata about the nodes/edges that you wish you could have to interpret your findings better?
  - We have data on each movie's revenue but did not represent it on the graph. If we make the size of the nodes correlates with their respective revenue, it will be easier to identify the most popular movie and interpret communities in the network more intuitively.