Finding Key Influential Actor in Movie Industry

BADS 7201 , 7202

Potchara	Vinitwattanakoon	6210412003
Ativit	Chaninchodeuk	6220412019

Outline

- Introduction
- Dataset
- Analysis
- Conclusion

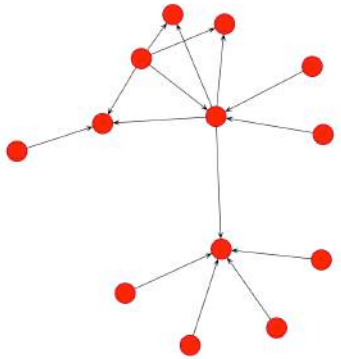
Introduction

Issues for film makers

Too many actors to choose from

Some Co-Lead actors has no chemistry

Difficultly choosing the right actors for the movie



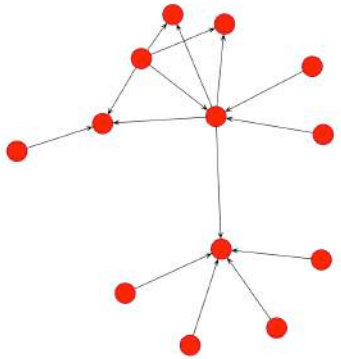
- Node = Actors
- Edge = Co-starring

Objective

Too many actors to choose from → Ranking actors

Some Co-Lead actors has no chemistry → Choose actors within same community

Difficultly choosing the right actors for the movie



- Node = Actors
- Edge = Co-starring

Process Flow

- **Data preprocessing**

  Data exploratory

  Filter data, transform data

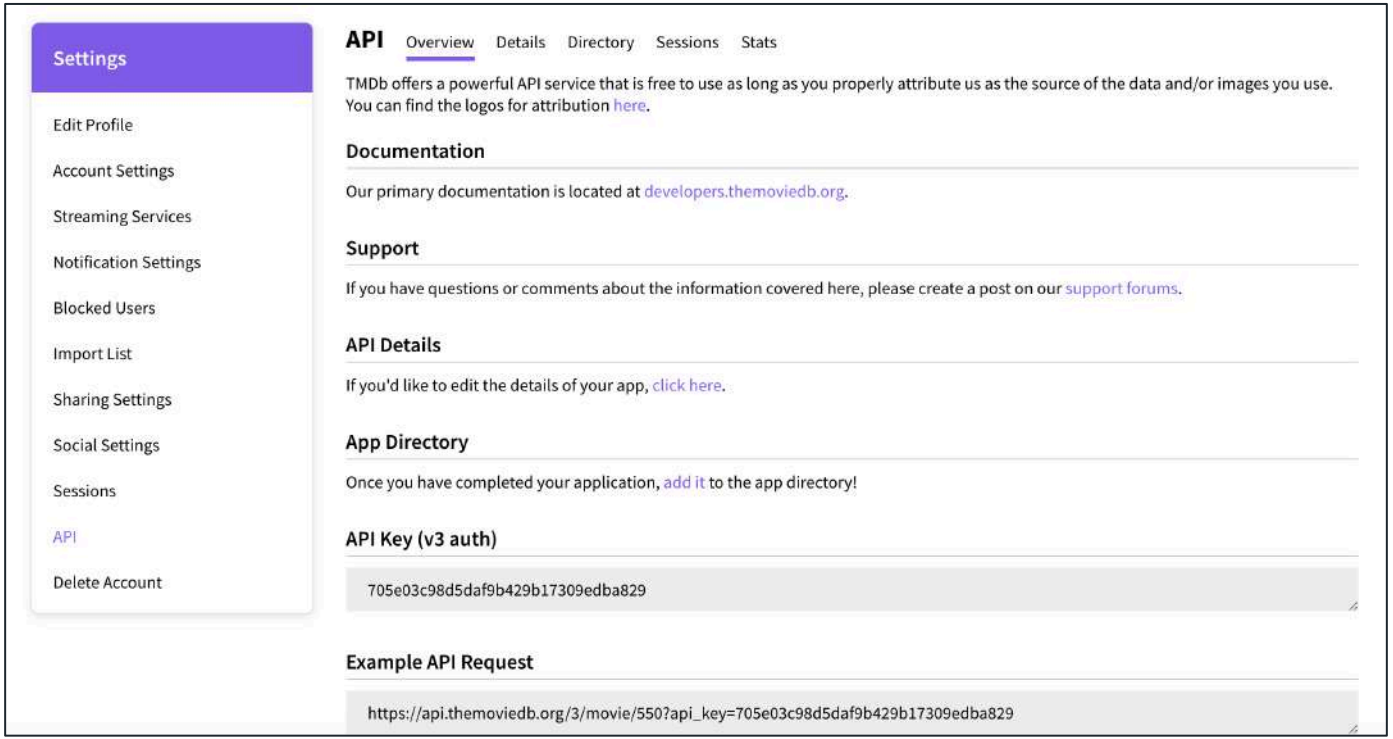
- **Create network graph**
- **Analysis**

  Centrality

  Community Detection

Dataset

TMDb API


Settings
Edit Profile
Account Settings
Streaming Services
Notification Settings
Blocked Users
Import List
Sharing Settings
Social Settings
Sessions
API
Delete Account
API
Overview
Details
Directory
Sessions
Stats
TMDb offers a powerful API service that is free to use as long as you properly attribute us as the source of the data and/or images you use.
You can find the logos for attribution here.
Documentation
Our primary documentation is located at developers.themoviedb.org.
Support
If you have questions or comments about the information covered here, please create a post on our support forums.
API Details
If you'd like to edit the details of your app, click here.
App Directory
Once you have completed your application, add it to the app directory!
API Key (v3 auth)
705e03c98d5daf9b429b17309edba829
Example API Request
https://api.themoviedb.org/3/movie/550?api_key=705e03c98d5daf9b429b17309edba829

TMDb API

- **tmdbsimple Version 2.6.6**
- **API Key**

  Register and verify an account to get API key

  API key = ‘705e03c98d5daf9b429b17309edba829’
- **Finding**

  Movie Dataset ： tmdb id, title, revenue, budget, product country, imdb rating, imdb votecount

  Crew Dataset ： crew id, name, department, job, character, popularity

Create Movie.csv

```python
import tmdbsimple as tmdb
import requests
import locale
import pandas as pd
imdb_id_list = []
tmdb_id_list = []
revenue_list = []
budget_list = []
title_list = []
product_country_list = []
vote_average_list = []
vote_count_list = []
Keyerror = []

def parsing(obj):
    mystr = ''
    j = 0
    for i in obj:
        if j != 0:
            mystr += ','
        mystr += i['iso_3166_1']
        j += 1

    return mystr
```

```python
for tmdb_id_lists in imdb_id :
  try :

    url = 'https://api.themoviedb.org/3/movie/'+ tmdb_id_lists
    +'?api_key=705e03c98d5daf9b429b17309edba829&language=en-US'


    print('fetching: ',tmdb_id_lists)


    response = requests.get(url)
    response_json = response.json()
    imdb_id = response_json["imdb_id"]
    tmdb_id = response_json["id"]
    revenue = response_json["revenue"]
    budget = response_json["budget"]
    title = response_json["title"]
    product_country = response_json['production_countries']
    vote_average = response_json['vote_average']
    vote_count = response_json['vote_count']


    product_country_str = parsing(product_country)
    imdb_id_list.append(imdb_id)
    tmdb_id_list.append(tmdb_id)
    revenue_list.append(revenue)
    budget_list.append(budget)
    title_list.append(title)
    product_country_list.append(product_country_str)
    vote_average_list.append(vote_average)
    vote_count_list.append(vote_count)

  except KeyError :
    print('KeyError----------------> ', tmdb_id_lists)
    C = Keyerror.append(tmdb_id_lists)
    pass
```

Jason Format


```
"production_companies": [
  {
    "id": 508,
    "logo_path": "/7PzJdsLGlR7oW4J0J5Xcd0pHGRg.png",
    "name": "Regency Enterprises",
    "origin_country": "US"
  },
  {
    "id": 711,
    "logo_path": null,
    "name": "Fox 2000 Pictures",
    "origin_country": ""
  },
  {
    "id": 20555,
    "logo_path": null,
    "name": "Taurus Film",
    "origin_country": ""
  },
  {
    "id": 54050,
    "logo_path": null,
    "name": "Linson Films",
    "origin_country": ""
  },
  {
    "id": 54051,
    "logo_path": null,
    "name": "Atman Entertainment",
    "origin_country": ""
  },
```

Create Movie.csv

	tmdb_id	title	product_country	startYear	imdb_votecount
0	400531.0	Frivolinas	ES	2014	15.0
1	11232.0	Kate & Leopold	US	2001	78160.0
2	602986.0	The Tango of the Widower and Its Distorting Mi...	CL	2020	76.0
3	299782.0	The Other Side of the Wind	FR,IR,US	2018	5995.0
4	339338.0	Toula, or the Genie of the Water	NE,DE	2017	25.0
...	...	...	...	...	...
159527	655187.0	Akelarre	AR,ES,FR	2020	97.0
159528	622812.0	The Secret of China	CN	2019	11.0
159529	NaN	NaN	NaN	2019	5.0
159530	NaN	NaN	NaN	2013	NaN
159531	NaN	NaN	NaN	2017	NaN

159532 rows × 5 columns

Create Crew.csv

```
import tmdbsimple as tmdb
import requests
import locale
import pandas as pd
id_list = []
imdb_id_list = []
name_list = []
popularity_list = []
Keyerror = []
Valueerror = []
Typeerror = []
count = 0

for crew_id in my_list :
  try :

    url = 'https://api.themoviedb.org/3/person/'+ crew_id
    +'?api_key=705e03c98d5daf9b429b17309edba829&language=en-US'

    count += 1
    print('fetching: ', crew_id ,'counting' , count)

    response = requests.get(url)
    response_json = response.json()
    id = response_json["id"]
    imdb_id = response_json["imdb_id"]
    name = response_json["name"]
    popularity = response_json["popularity"]
    gender = response_json["gender"]

    imdb_id_list.append(imdb_id)
    id_list.append(id)
    name_list.append(name)
    popularity_list.append(popularity)
    gender_list.append(gender)

  except KeyError :
    print('KeyError----------------> ', crew_id)
    C = Keyerror.append(crew_id)
    pass
```

Create Crew.csv

	movie_id	crew_id	name	department	job	character	popularity
0	11232	5344	Meg Ryan	Acting	Actor	Kate McKay	8.368
1	11232	6968	Hugh Jackman	Acting	Actor	Leopold	17.615
2	11232	23626	Liev Schreiber	Acting	Actor	Stuart Besser	7.494
3	11232	33654	Breckin Meyer	Acting	Actor	Charlie McKay	7.794
4	11232	10871	Natasha Lyonne	Acting	Actor	Darci	5.356
...	...	...	...	...	...	...	...
125134	615177	11677	Terry Chen	Acting	Actor	Officer Steve Choi	3.247
125135	615177	62709	Garfield Wilson	Acting	Actor	Steve (Angry Man)	1.098
125136	615177	1406026	Arielle Tuliao	Acting	Actor	Sharon	1.473
125137	615177	228893	Jag Bal	Acting	Actor	Snarky IT Guy	1.400
125138	615177	1904663	Sonia Sunger	Acting	Actor	Présentatrice du journal télévisé	0.600

125139 rows × 7 columns

Dataset

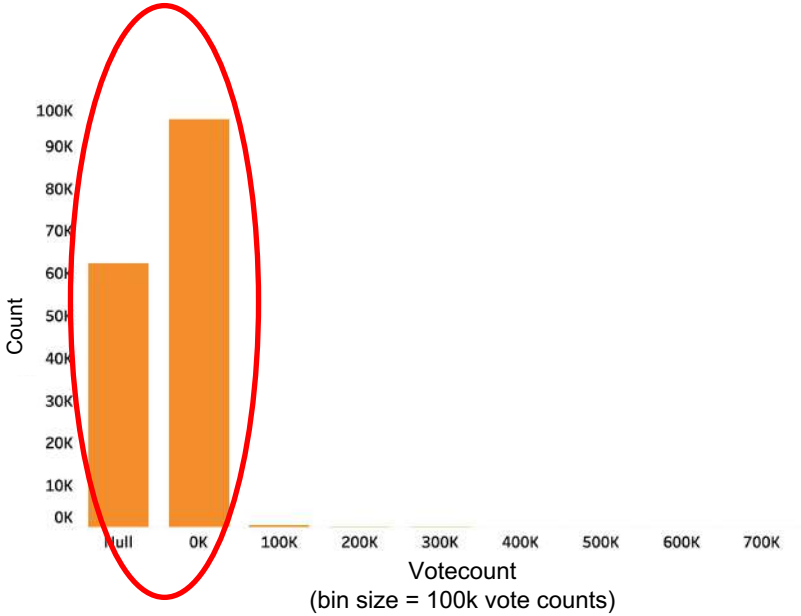
movies dataset
Movie.csv

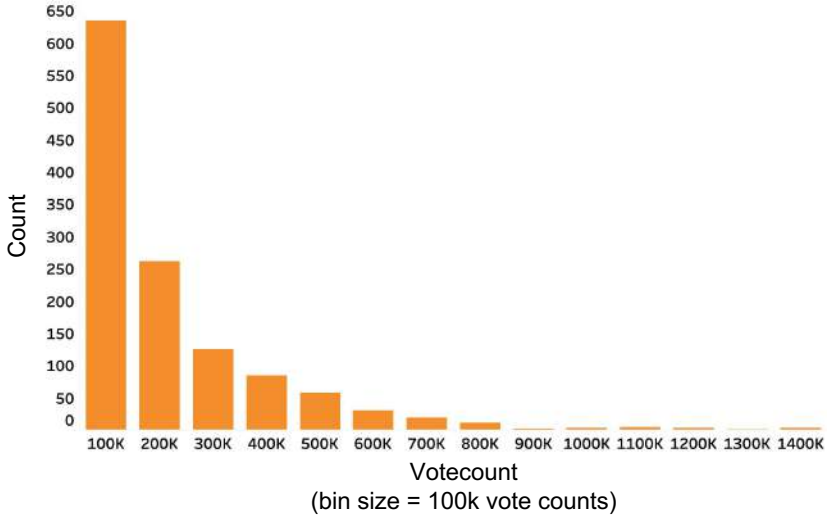
	tmdb_id	title	product_country	startYear	imdb_votecount
0	400531.0	Frivolinas	ES	2014	15.0
1	11232.0	Kate & Leopold	US	2001	78160.0
2	602986.0	The Tango of the Widower and Its Distorting Mi...	CL	2020	76.0
3	299782.0	The Other Side of the Wind	FR,IR,US	2018	5995.0
4	339338.0	Toula, or the Genie of the Water	NE,DE	2017	25.0

actors dataset
Crew.csv

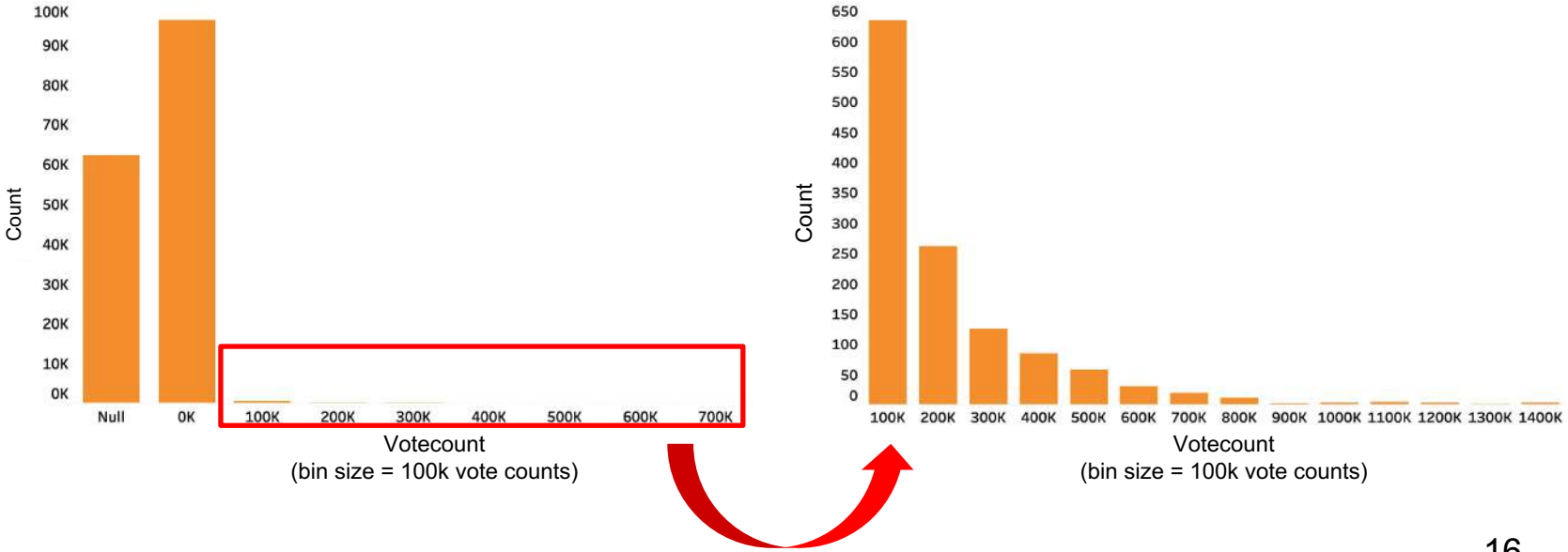
	movie_id	crew_id	name	department	job	character	popularity
1311	503919	10965	Chris Columbus	Production	Executive Producer	NaN	2.871
7114	503919	5293	Willem Dafoe	Acting	Actor	Thomas Wake	15.189
13884	503919	40142	Robert Fernandez	Sound	Sound Re-Recording Mixer	NaN	0.600
31106	503919	376	Arnon Milchan	Production	Executive Producer	NaN	1.702
31791	503919	1197157	Paul Rutledge	Crew	Stunts	NaN	1.380
32006	503919	11288	Robert Pattinson	Acting	Actor	Thomas Howard	11.369
32351	503919	11183	Eleanor Columbus	Production	Executive Producer	NaN	2.407

Exploratory Data Analysis

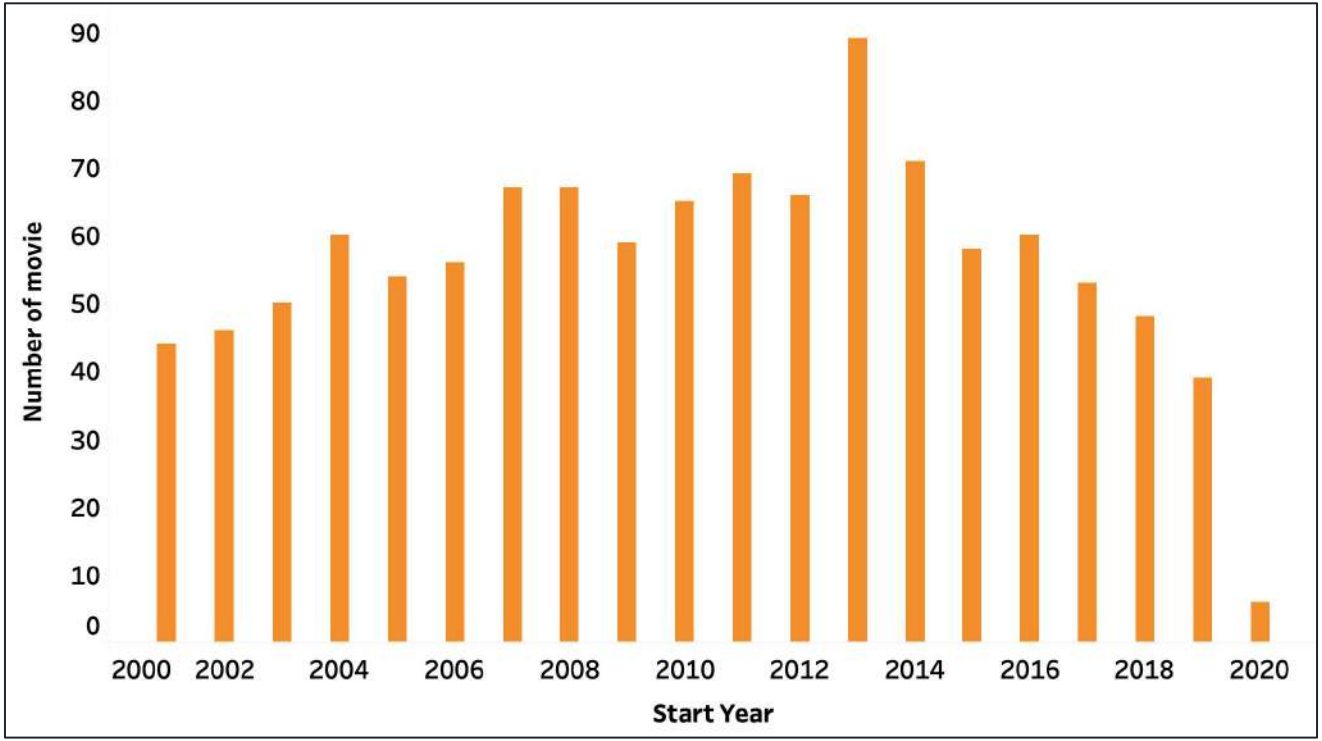




Exploratory Data Analysis



Exploratory Data Analysis



What movies are filtered?

- **Vote Count**

  Select a vote count of movie that has more than 100,000 of total user votes.

- **Product Country**

  Only select produced in US.

Too many actors

The Lord Of The Rings
The Fellowship Of The Ring.

	name	popularity
139	Elijah Wood	13.101
140	Ian McKellen	13.451
141	Liv Tyler	7.959
142	Viggo Mortensen	10.152
143	Sean Astin	8.999
144	Cate Blanchett	12.751
145	John Rhys-Davies	7.129
146	Billy Boyd	2.978
147	Dominic Monaghan	7.423
148	Orlando Bloom	6.675
149	Christopher Lee	10.467
150	Hugo Weaving	6.959
151	Sean Bean	22.236
152	Ian Holm	4.658
153	Andy Serkis	8.840

244	Tim McLachlan	1.013
245	Liz Merton	0.600
246	Arnold Montey	0.994
247	Dean Morganty	0.694
248	Greg Morrison	0.600
249	Blair Morton	0.600
250	Andrew Munro	0.728
251	David J. Muzzerall	0.600
252	Shane Rangi	2.588
253	Chris Reid	0.600
254	Steve Reinsfield	0.600
255	Larry Rew	0.600
256	Grant Roa	1.220
257	Thomas Robins	0.694
258	Vincent Roxburgh	0.600
259	Chris Ryan	0.600
260	Paul Shapcott	0.600
261	Samuel E. Shore	0.840

Total Number of Actors = 131

Too many actors

	movie_id	crew_id	name	department	job	character	popularity
1311	503919	10965	Chris Columbus	Production	Executive Producer	NaN	2.871
7114	503919	5293	Willem Dafoe	Acting	Actor	Thomas Wake	15.189
13884	503919	40142	Robert Fernandez	Sound	Sound Re-Recording Mixer	NaN	0.600
31106	503919	376	Arnon Milchan	Production	Executive Producer	NaN	1.702
31791	503919	1197157	Paul Rutledge	Crew	Stunts	NaN	1.380
32006	503919	11288	Robert Pattinson	Acting	Actor	Thomas Howard	11.369
32351	503919	11183	Eleanor Columbus	Production	Executive Producer	NaN	2.407

The Lord Of The Rings
The Fellowship Of The Ring.

What actors are filtered?

- **Popularity of actors**

  Select actor with a popularity value of more than or equal to 7

- **Job**

  Only select job = actor on Crew.csv

Dataset

- **Original data base on periods 20 years.**

  159,532 movies

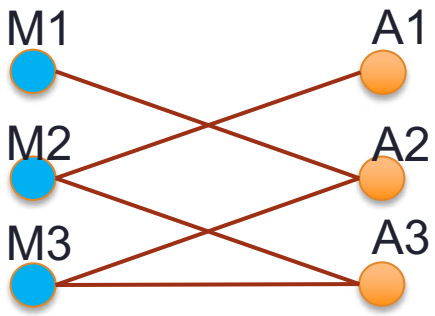
  125,139 actors

- **Filtered (Product Country, Vote count, Popularity)**

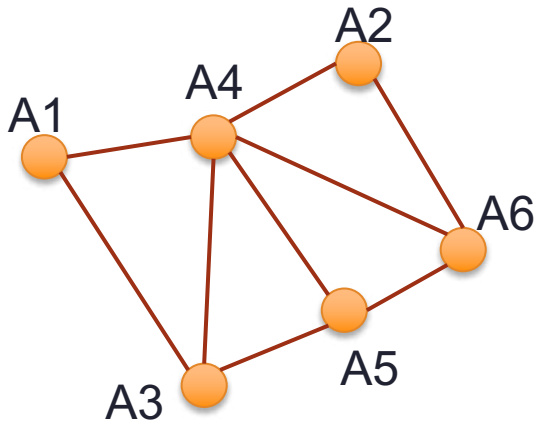
  868 movies

  1,091 actors

Data preprocessing, data tranformation



movie_id	crew_id	name	character
49026	3894	Christian Bale	Bruce Wayne / Batman
49026	3895	Michael Caine	Alfred Pennyworth
49026	64	Gary Oldman	Commissioner James Gordon
49026	1813	Anne Hathaway	Selina Kyle / Catwoman
49026	2524	Tom Hardy	Bane
49026	8293	Marion Cotillard	Miranda Tate / Talia al Ghul
49026	24045	Joseph Gordon-Levitt	John Blake



actor1	actor2	name1	name2
11670	24000	Christian Bale	Heath Ledger
11670	43620	Christian Bale	Michael Caine
11670	21719	Christian Bale	Gary Oldman
11670	39936	Christian Bale	Maggie Gyllenhaal
11670	45622	Christian Bale	Morgan Freeman

Convert Bipartite to Unipartite Graph

```python
#create [movie-actor] object
mov_actor = df.groupby(['movie'])['actor_1'].apply(','.join)
mov_actor
```

```
movie
0                                    31209,34569,65181
1                            62656,39289,15026,44008,11159
2                            26925,26237,49865,39159,30826
3                                  29526,34012,35906,18593
4                                   31209,34569,6929,46072
                                  ...
3532    11379,3048,61695,36637,12194,34084,26129,32054
3533                                 10867,11220,23840
3541                                   152,8879,36313
3545                                            54887
3551                           52866,18722,43620,12747
Name: actor_1, Length: 1091, dtype: object
```

```python
#find combination between 2 pair of actors from [mov-actor] object
mylist = []
for i in mov_actor:
    actor_list = i.split(',')
    comb = combinations(actor_list, 2)
    for i in list(comb):
        #convert combination inside into integer so it can be sorted
        edge_pair = list(map(int, i))
        mylist.append(sorted(edge_pair))
```

	actor1	actor2
0	31209	34569
1	31209	65181
2	34569	65181
3	39289	62656
4	15026	62656

Create weight from sum of repeated edge pairs

```
#create ['repeat'] col that will be used as weight of edges
temp = df_merged2.groupby('concat').agg({'concat':'size'}).rename(columns={'name':'repeat'})
temp.columns = ['repeat']

print(temp.shape)
temp.sort_values('repeat', ascending=False)
```

```
(10801, 1)
```

concat	repeat
(Chris Evans, Stan Lee)	13
(Robert Downey Jr., Stan Lee)	10
(Samuel L. Jackson, Stan Lee)	10
(Scarlett Johansson, Stan Lee)	9
(Alan Rickman, Robbie Coltrane)	8
...	...
(David Duchovny, Vince Vaughn)	1
(David Duchovny, Winona Ryder)	1

Create weight from sum of repeated edge pairs

Weight

	actor1	actor2	name1	name2	concat	repeat	ebunch
1777	11379	57874	Chris Evans	Stan Lee	(Chris Evans, Stan Lee)	13	[Chris Evans, Stan Lee, 13]
1713	52680	57874	Robert Downey Jr.	Stan Lee	(Robert Downey Jr., Stan Lee)	10	[Robert Downey Jr., Stan Lee, 10]
1733	54954	57874	Samuel L. Jackson	Stan Lee	(Samuel L. Jackson, Stan Lee)	10	[Samuel L. Jackson, Stan Lee, 10]
3637	55565	57874	Scarlett Johansson	Stan Lee	(Scarlett Johansson, Stan Lee)	9	[Scarlett Johansson, Stan Lee, 9]
958	52549	54022	Robbie Coltrane	Rupert Grint	(Robbie Coltrane, Rupert Grint)	8	[Robbie Coltrane, Rupert Grint, 8]
...	...	...	...	...	...	...	...
4033	24198	29526	Helena Bonham Carter	Jim Carrey	(Helena Bonham Carter, Jim Carrey)	1	[Helena Bonham Carter, Jim Carrey, 1]
4034	17795	29526	Dustin Hoffman	Jim Carrey	(Dustin Hoffman, Jim Carrey)	1	[Dustin Hoffman, Jim Carrey, 1]
4035	27132	29526	Jane Lynch	Jim Carrey	(Jane Lynch, Jim Carrey)	1	[Jane Lynch, Jim Carrey, 1]
4036	7007	43388	Billy Connolly	Meryl Streep	(Billy Connolly, Meryl Streep)	1	[Billy Connolly, Meryl Streep, 1]
10800	12747	43620	Clémence Poésy	Michael Caine	(Clémence Poésy, Michael Caine)	1	[Clémence Poésy, Michael Caine, 1]

10801 rows × 7 columns

Create networkx graph

```
[47] #create function to pass the edge pair and weight into nx.add_edges_from
     def edge_generator(reader):
         reader
         for row in reader:
             row[2] = float(row[2]) # convert numeric weight to float
             yield (row[0],
                    row[1],
                    dict(weight=row[2]))
```

```
[48] #create undirected graph
     G = nx.Graph()

     #add nodes
     G.add_nodes_from(actor_table.name.tolist())

     #add edge pairs with weight from edge_generator function
     G.add_edges_from(edge_generator(df_edge.ebunch.tolist()))
```

	actor	name
0	84	Aaron Eckhart
1	152	Aaron Paul
2	214	Abbie Cornish
3	282	Abigail Breslin
4	548	Adam Sandler

Graph Information

```
print(nx.info(G))

Is G graph connected:  True
Name:
Type: Graph
Number of nodes: 864
Number of edges: 10800
Average degree:  25.0000
```

Actor

Network Graph


Vin Diesel


Jeremy Renner


Jon Favreau


Cobie Smulders


Idris Elba


Robert Downey Jr


Sebastian Stan


Hayley Atwell


Stan lee


Chris Hemsworth


Benedict Cumberbatch


Gwyneth Paltrow


Samuel L. Jackson


Chris Evans


Scarlett Johansson


Tom Hiddleston

4 6 5 7 5 7 5 10 5 6 7 8 5 6 7 5 10 5 9 7 13 7 5 7 7 5

Network Graph Analysis

Centrality

Centrality Concept

Centrality	Main idea
Degree Centrality	Ratio number of degree link of a node
Closeness Centrality	Ratio of path length of the shortest path between the node and all other nodes (Euclidean Distance)
Betweenness Centrality	Ratio number of times a node acts as a bridge along the shortest path between two other nodes

Centrality Concept

Centrality	Main idea
Degree Centrality	Ratio number of degree link of a node
Closeness Centrality	Ratio of path length of the shortest path between the node and all other nodes (Euclidean Distance)
Betweenness Centrality	Ratio number of times a node acts as a bridge along the shortest path between two other nodes

Betweenness Centrality

$$C_B(v) = \sum_{s,t \in V} \frac{\sigma(s,t|v)}{\sigma(s,t)}$$

V is a set of vertex

$\sigma(s,t)$ = total number of shortest part between node s and t

$\sigma(s,t|v)$ = total number of shortest part between node s and t that pass through v

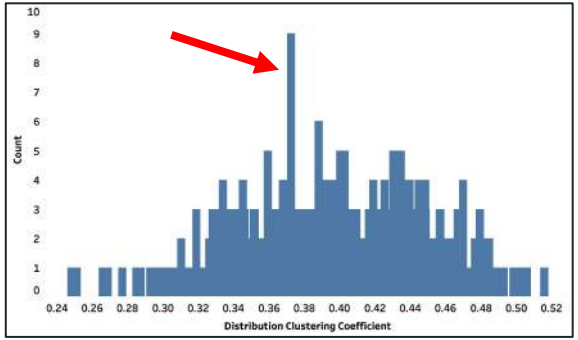
Betweenness Centrality Score

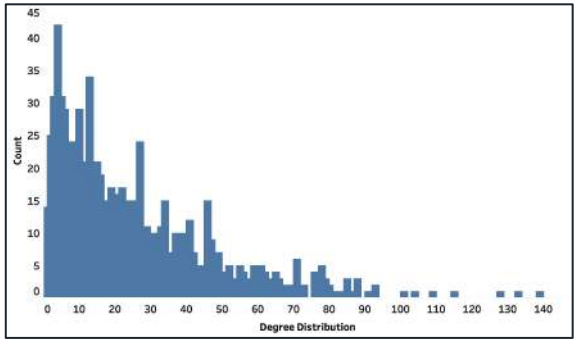
```
#compute betweenness centrality
btw = nx.betweenness_centrality(G,k=100)
dict_btw = dict(btw)
```

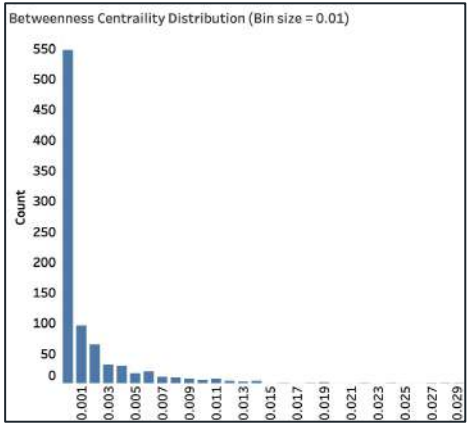
Name	Betweenness Centrality Score
Samuel L. Jackson	0.0317
Hugh Jackman	0.0265
Natalie Portman	0.0248
Liam Neeson	0.0248
Dwayne Johnson	0.0206
Stan Lee	0.0189
Ryan Reynolds	0.0182
Jason Statham	0.0165
Tilda Swinton	0.0165
Scarlett Johansson	0.0157
Vince Vaughn	0.0152
Bruce Willis	0.0143
Matt Damon	0.0139
Ben Affleck	0.0138
Channing Tatum	0.0132
Ben Stiller	0.0130
Jason Bateman	0.0126
Jastin Theroux	0.0120
Josh Brolin	0.0117
Mark Wahlberg	0.0115

Node Distribution:

Degree, Clustering Coefficient, Betweenness Centrality







	degree	clust_coeff	btw_cent
max	139	0.52	0.0298
min	1	0.25	0.0000
mean	25	0.39	0.0019
std	22	0.05	0.0035

Community Detection

Louvain Algorithm

$$\Delta Q(i \longrightarrow C) = \left[\frac{\sum_{in} + k_i, in}{2m} - \left(\frac{\sum_{tot} + k_i}{2m}\right)^2\right] - \left[\frac{\sum in}{2m} - \left(\frac{\sum tot}{2m}\right)^2 - \left(\frac{k_i}{2m}\right)^2\right]$$

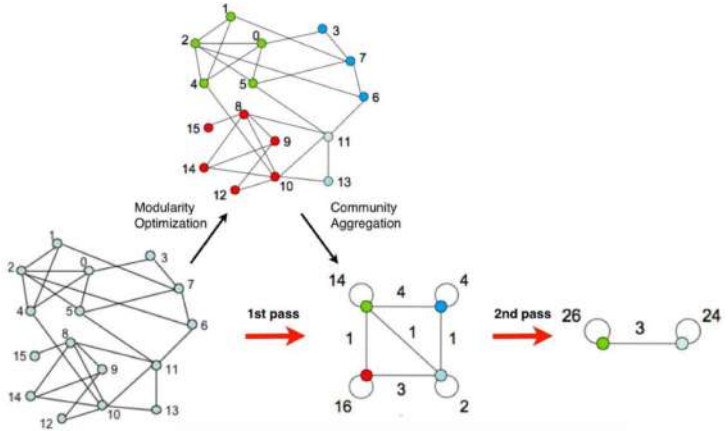
$\sum in$ = sum of link weights between nodes in C

$\sum tot$ = sum of all link weights of nodes in C

k_i, in = sum of link weights between node i and C

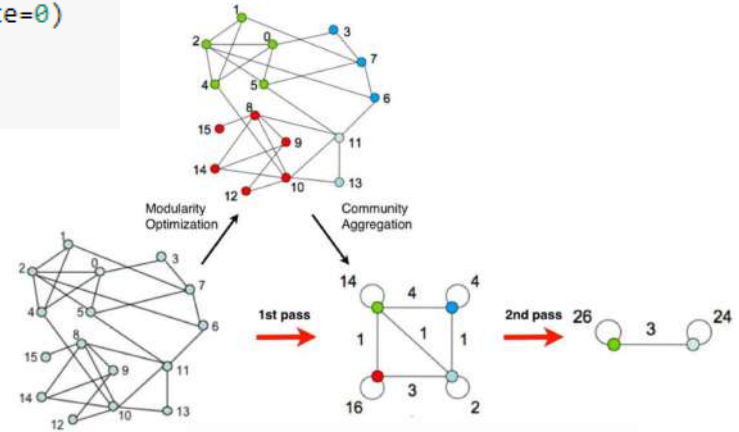
k_i = sum of all link weights of node i

$$\Delta Q = \Delta Q(i \longrightarrow C) + \Delta Q(D \longrightarrow i)$$



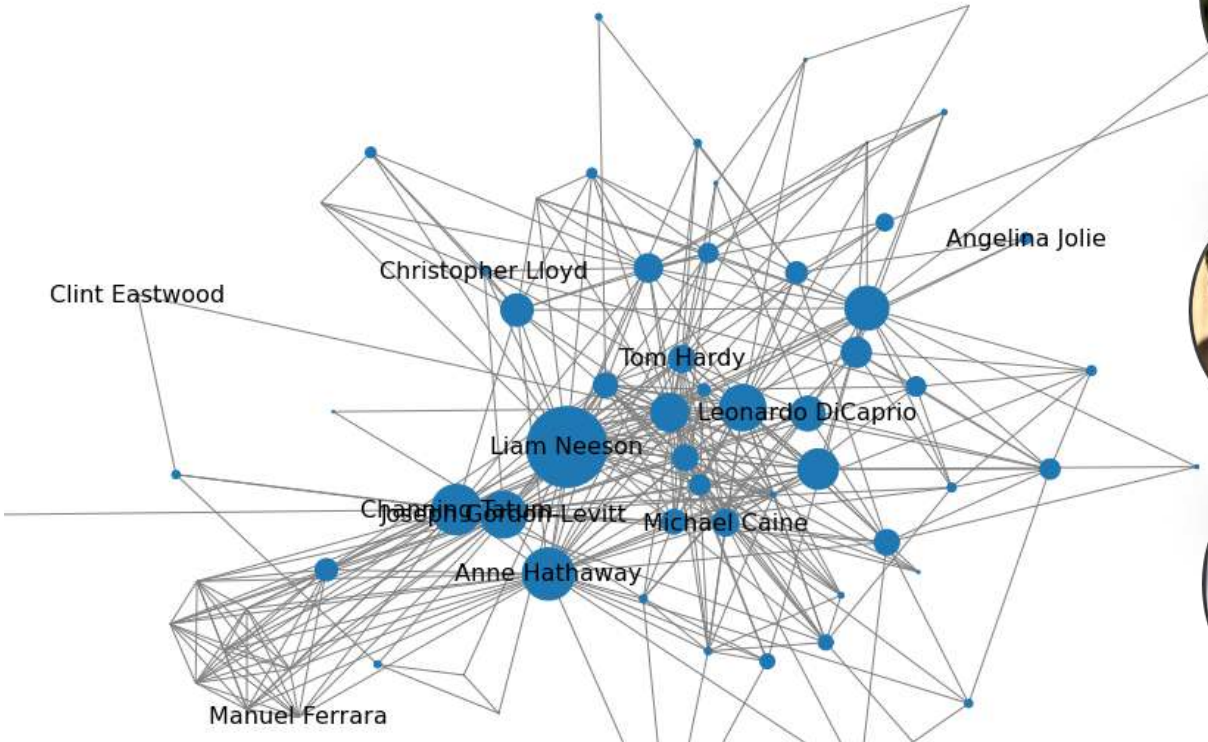
Louvain Algorithm

```
[43] #find community using Louvain Algorithm
     partition = community.best_partition(G, random_state=0)
     #create community dictionary
     dict_comm = dict(partition)
```



“All Stars”

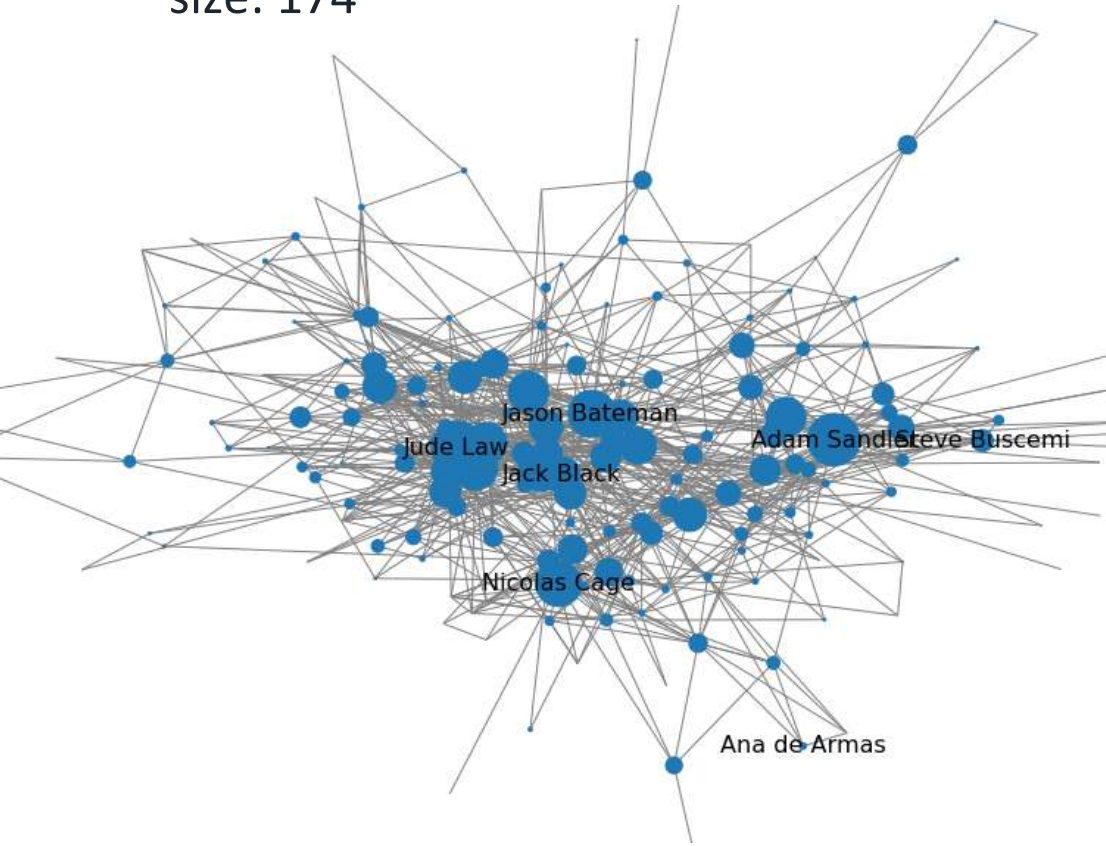
Community group 0
size: 71



“Support & Comedy Stars”

Community group 1

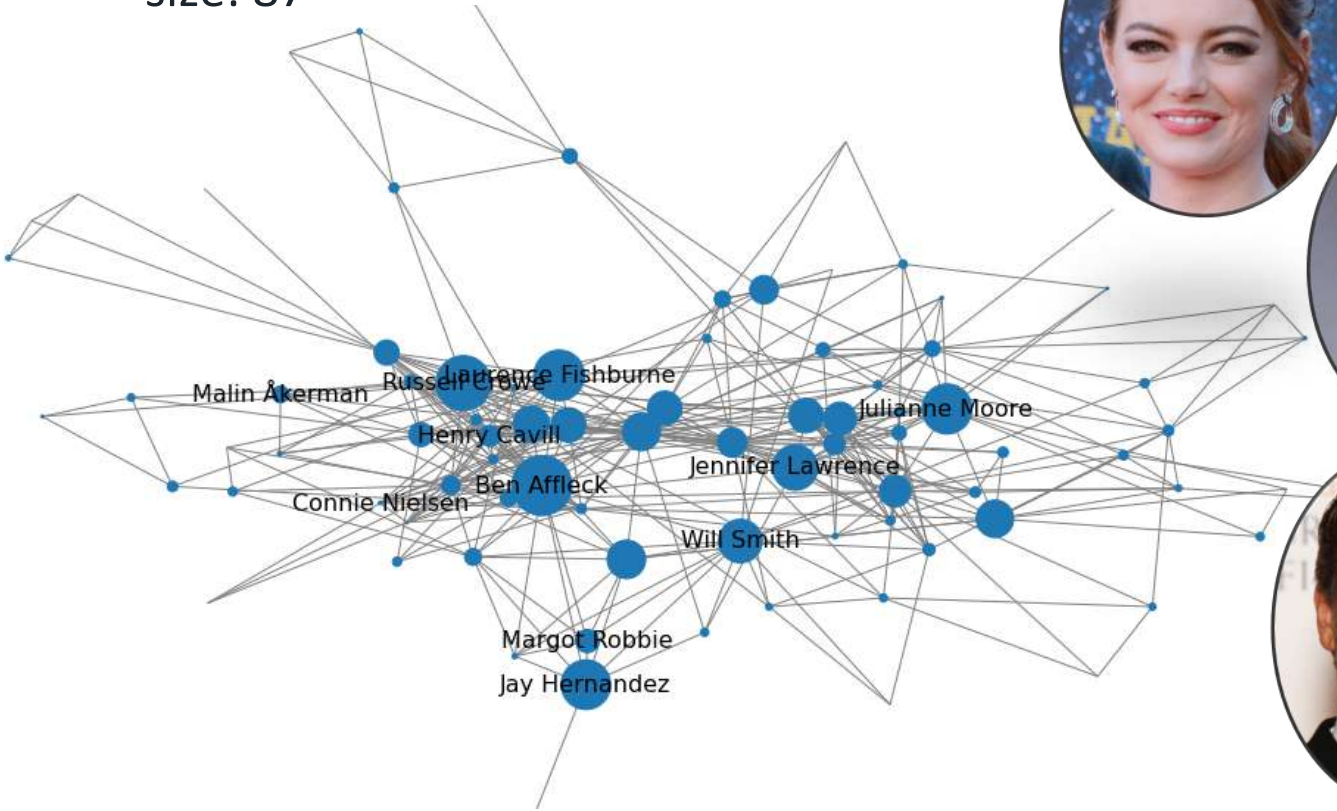
size: 174

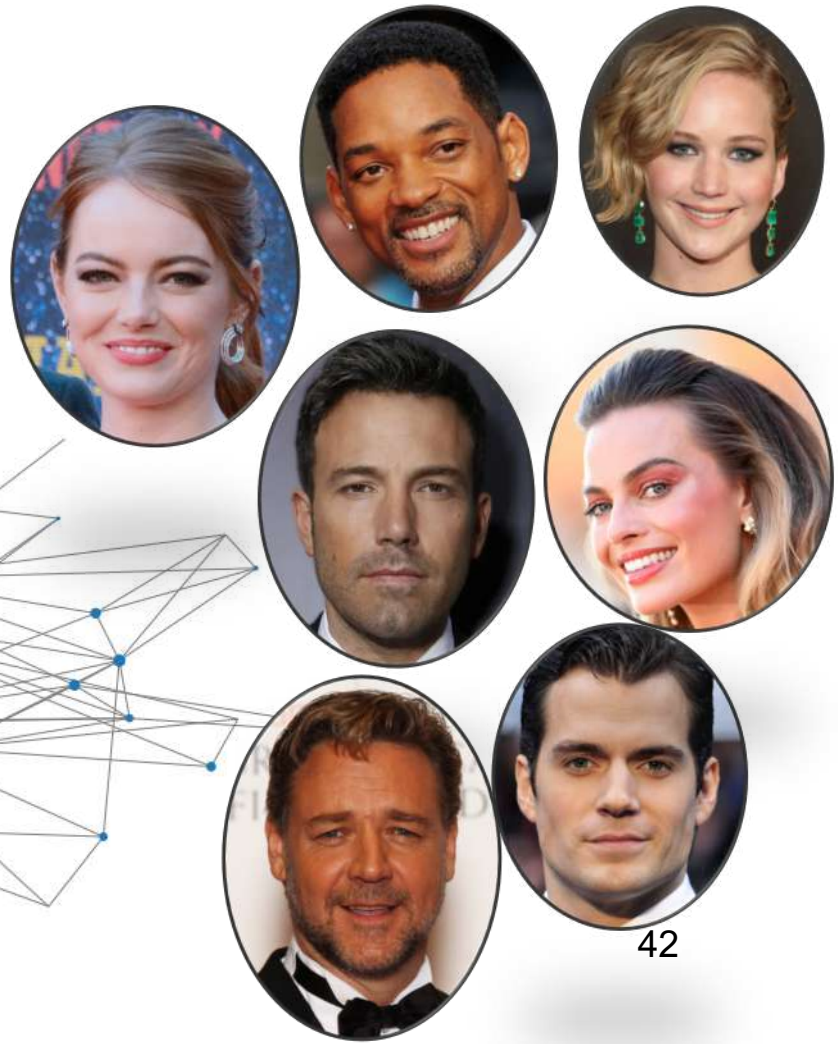


“Lead Stars 1”

Community group 2

size: 87

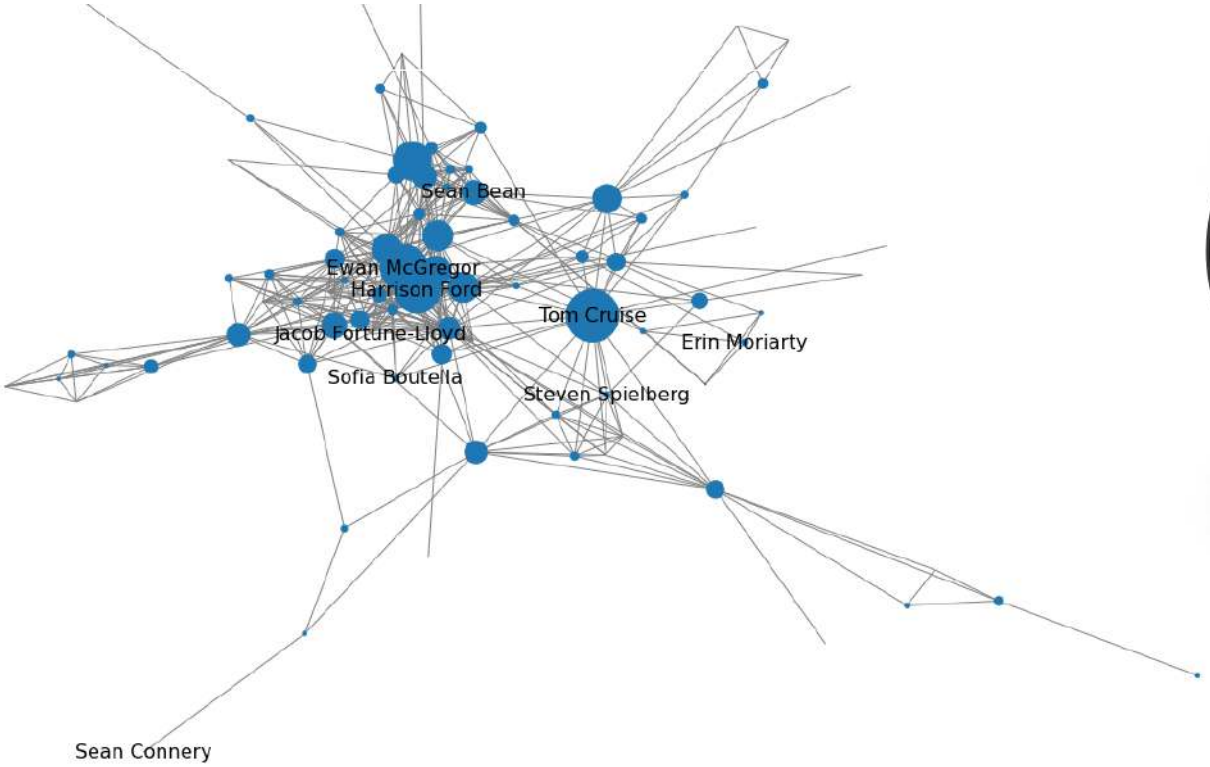




“Old Inactive Stars”

Community group 3

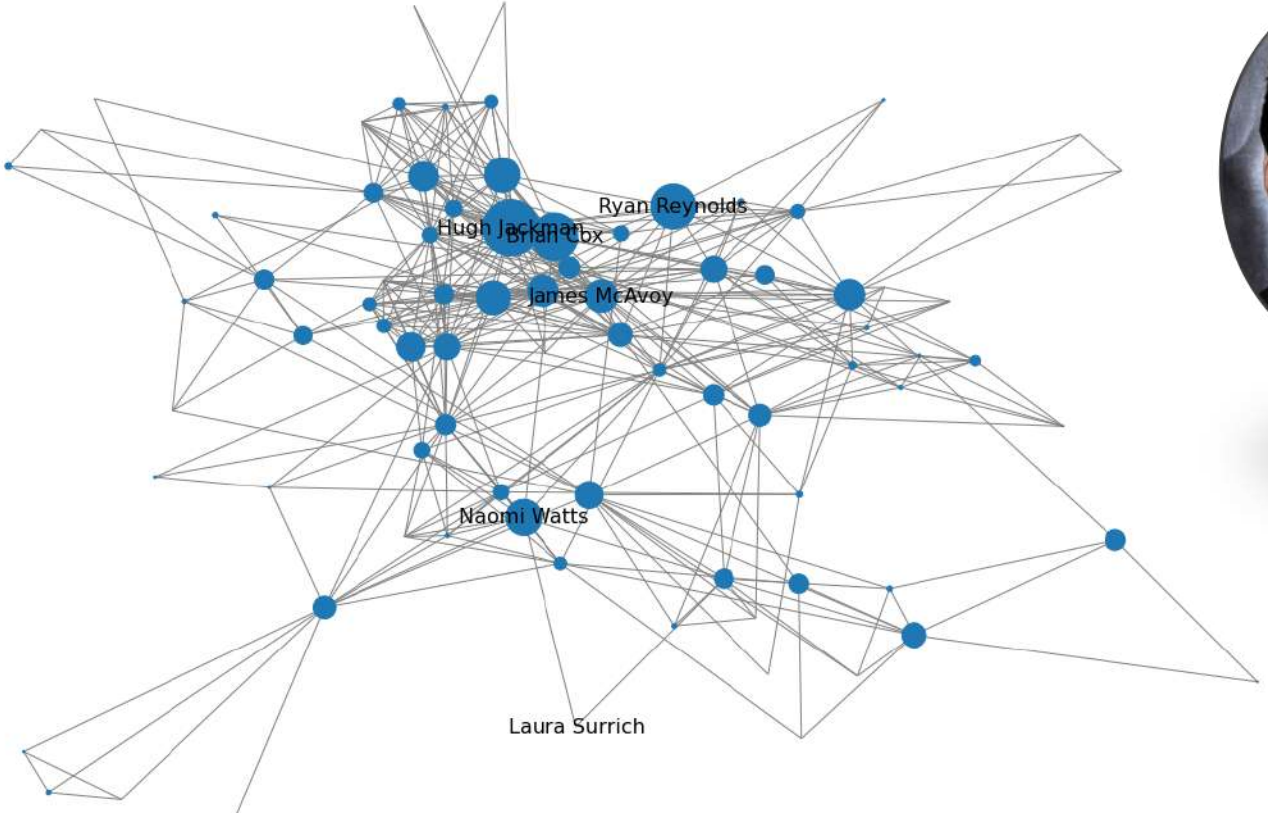
size: 84



“Fox Movie (X-men) Stars”

Community group 4

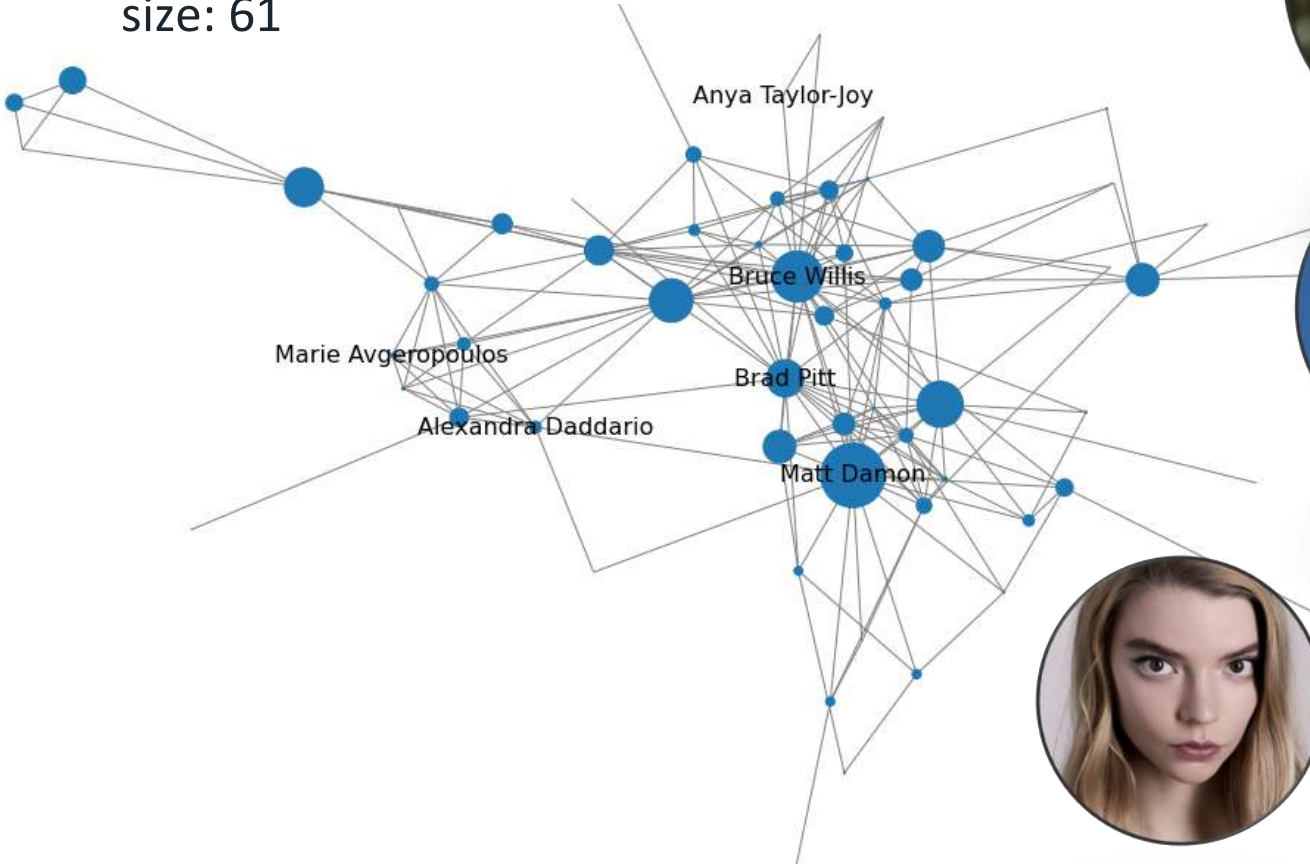
size: 82



“Lead Stars 2”

Community group 5

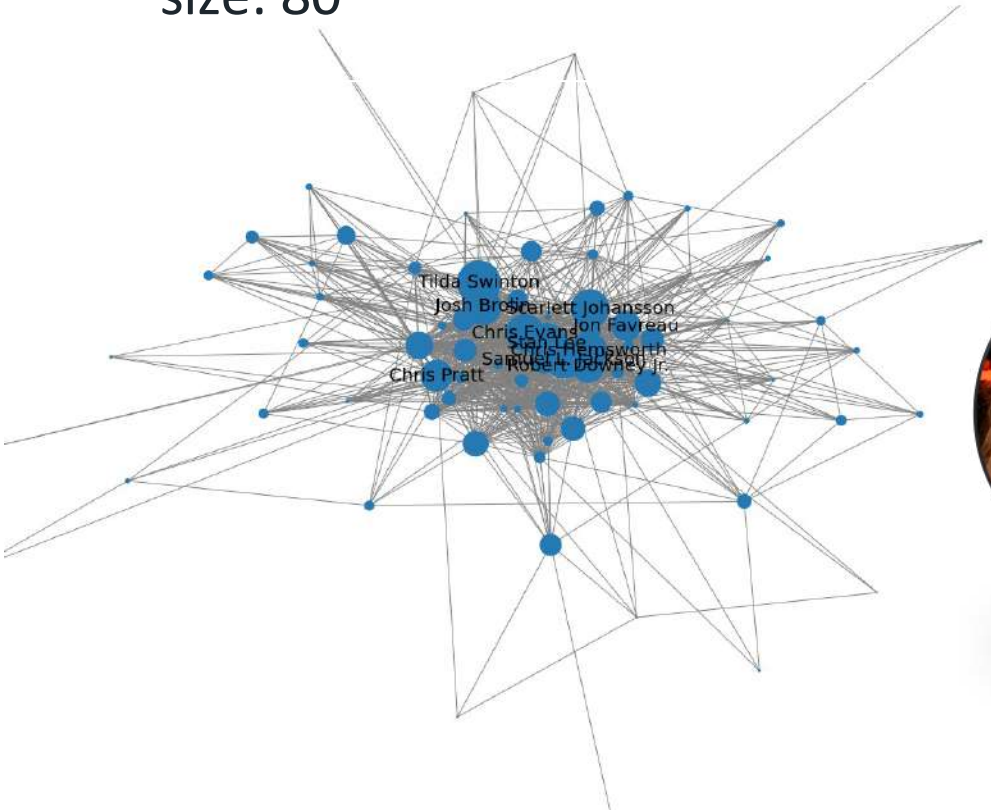
size: 61



“Marvel’s Avenger Stars”

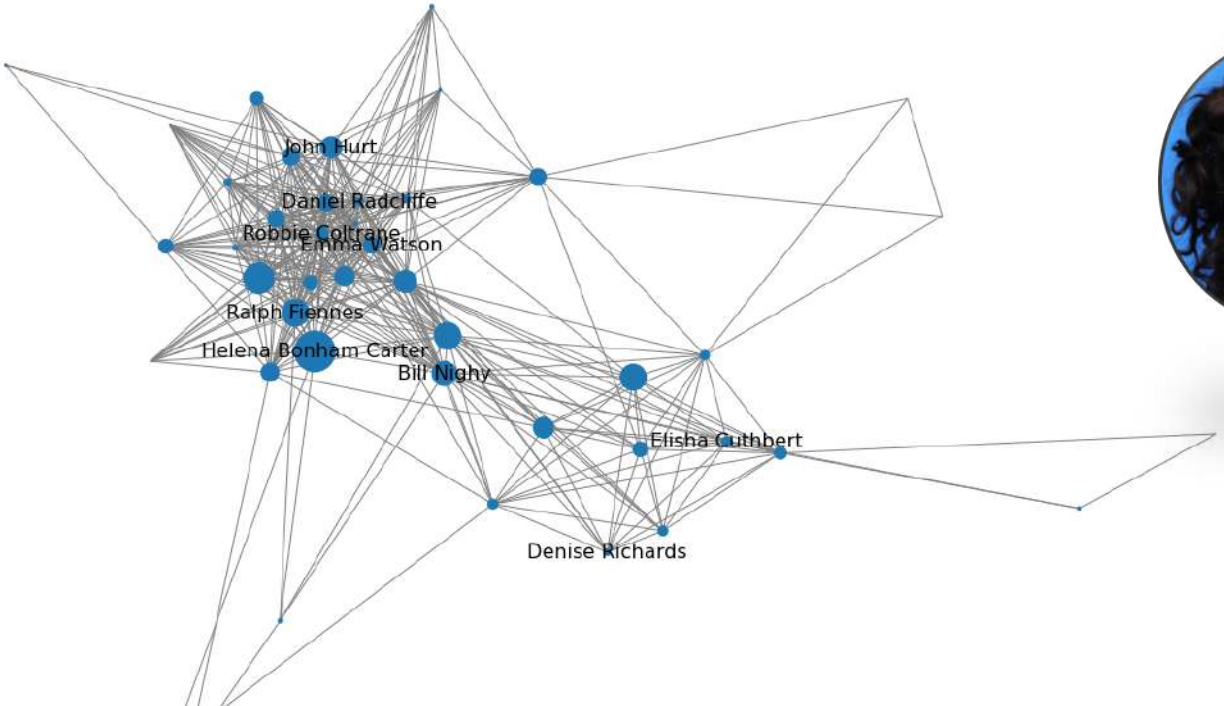
Community group 6

size: 80



“English Stars”

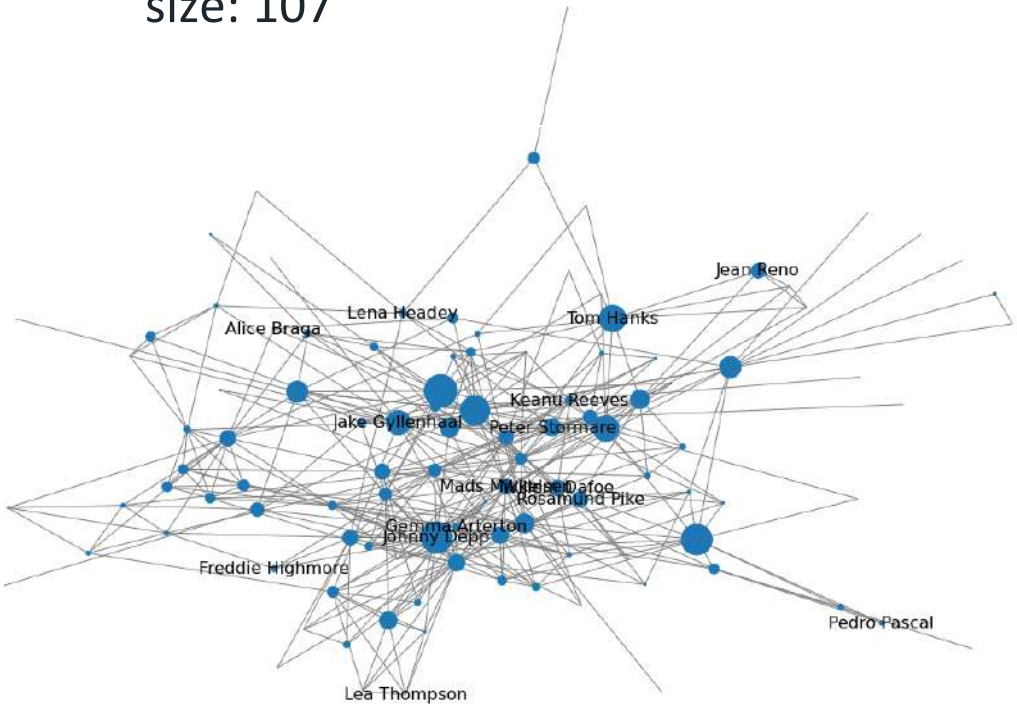
Community group 7
size: 44



“Well-known Niche Stars”

Community group 8

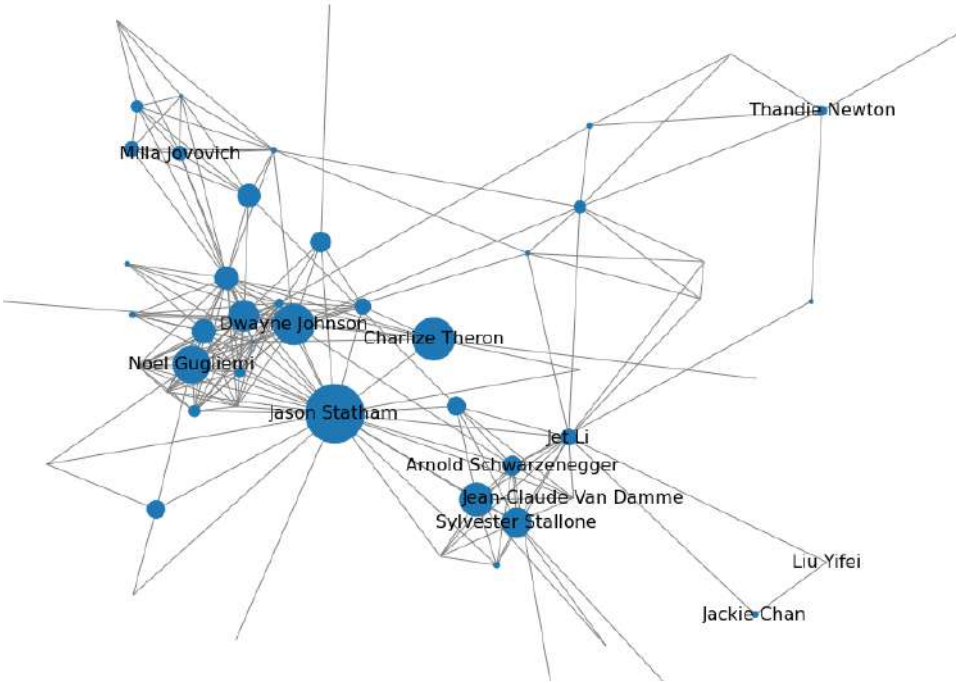
size: 107



“Action Stars”

Community group 9

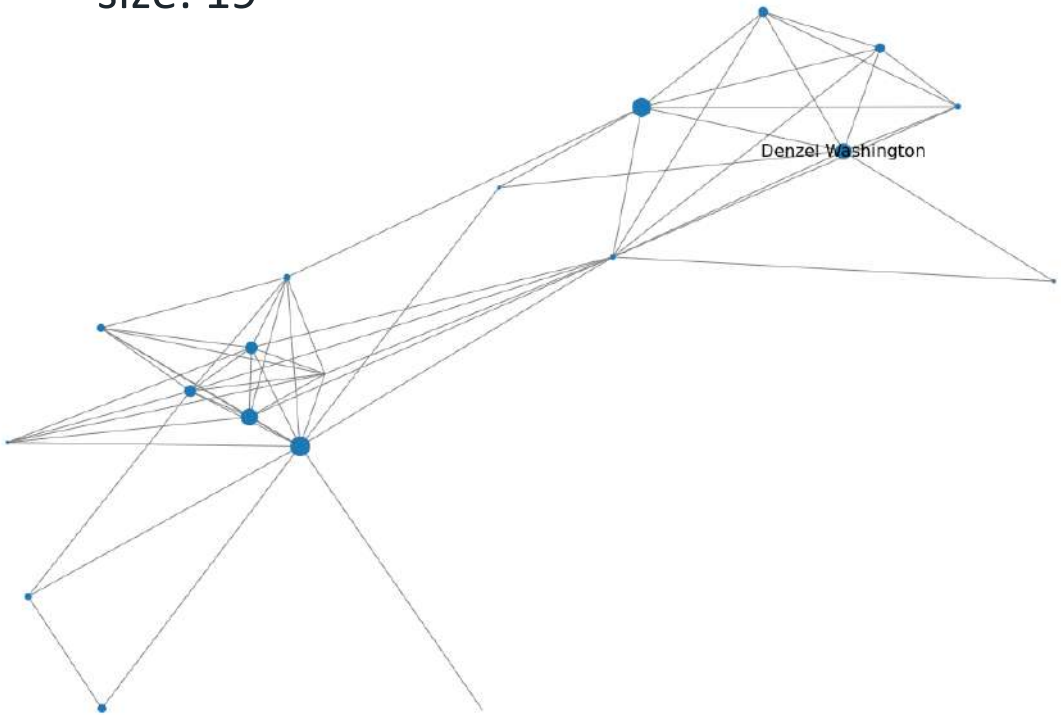
size: 55



“Niche Stars”

Community group 10

size: 19



Summary Between Community

		Mean			
Community Group	Gr. Name	Amount Movie Played	Popularity	degree	btw_cent
0	All Stars	12	21	52	0.0062
1	Support Actors + Comedy	6	15	23	0.0031
2	Lead Actor/Actress 1	10	19	39	0.0048
3	Old Inactive Actors	7	19	34	0.0043
4	Fox Moive (X-men)	10	17	41	0.0061
5	Lead Actor/Actress 2	8	23	27	0.0022
6	Avenger	17	20	85	0.0095
7	English Actors	10	17	42	0.0018
8	Well-known Niche Actors	10	23	34	0.0029
9	Action Stars	8	24	27	0.0055
10	Niche Actors	8	11	23	0.0010

Summary Between Community

		Mean			
Community Group	Gr. Name	Amount Movie Played	Popularity	degree	btw_cent
0	All Stars	12	21	52	0.0062
1	Support Actors + Comedy	6	15	23	0.0031
2	Lead Actor/Actress 1	10	19	39	0.0048
3	Old Inactive Actors	7	19	34	0.0043
4	Fox Moive (X-men)	10	17	41	0.0061
5	Lead Actor/Actress 2	8	23	27	0.0022
6	Avenger	17	20	85	0.0095
7	English Actors	10	17	42	0.0018
8	Well-known Niche Actors	10	23	34	0.0029
9	Action Stars	8	24	27	0.0055
10	Niche Actors	8	11	23	0.0010

Conclusion

Centrality (Betweenness)

- Tell the versatility of actors and how likely they will accept role having to work with different groups of actors

Community

- Help understand the behavior of accepting roles in movie of each actors

Limitation

- Could not cluster in a group of genre based
- The model does not take into account of the presence (screen times) of actors within movie

THANKS!