Hollywood Actors Analysis

Project Final Report - LE/EECS 4414/5414

Usama Abdali
usama01@my.yorku.ca
York University

Aggrim Arora
arora10@my.yorku.ca
York University

Arnesh Dawar
arneshie@my.yorku.ca
York University

Manpreet Saini
msaini12@my.yorku.ca
York University

ABSTRACT

Network analysis is the process of gathering large data sets and converting them into a network composed of nodes and edges. From this network, information such as link prediction, communities, and centrality can be obtained. We noticed how massive and popular the movie industry is and decided to create a network around the Hollywood movie industry. We then decided to create a network consisting of Hollywood actors, where two actors are linked together if they appeared in the same movie. With our network, we hoped to answer a few questions including, predicting whether certain actors would fit great in a specific movie, what type of community actors fall under, and which actors are the most central. We used several methods to answer these questions including, the Girvan-Newman community detection algorithm, Degree centrality, and the Preferential Attachment Algorithm for link prediction. We ended up obtaining a list of the most central actors, found communities, and predicted future movie collaborations.

1 INTRODUCTION/MOTIVATION

With Hollywood movies nowadays grossing millions, if not, billions of dollars, it really puts into perspective just how big the Hollywood film industry really is. When we look at the salary of the top actors, it makes us wonder why exactly they are worth this much and if there is a way to visualize this. As a result of this curiosity, we decided to create a network that consisted of Hollywood actors and the movies they appeared in. So we had the actors be represented as nodes, with an edge between two nodes if they appeared in the same film. The edges also have a weight that represents the number of movies done between two actors (nodes).The network we created allowed us to perform many different types of analysis and answer various network related questions.

1.1 Goal

Link prediction is a way to predict whether two unconnected nodes will connect in the future, so for our network, it would be to predict whether two actors, who haven't acted in the same film, would act together in the future. When performing a network analysis on link prediction, we want to know what the probability of an edge forming between two actors who have not acted in a film together is, as well as what the weight of the edge would be. This type of analysis would tell us what pairs of actors would be a great fit for a future movie given the genre.

A community is a subset of nodes within a graph such that the connections between the nodes are denser than the rest of the network. We wanted to be able to visualize the various communities being generated as it would help us answer, which communities do actors lie in based on the movies they acted in? This information would also show us whether there are dominant communities or if all the communities are evenly distributed.

Eigenvector centrality is the measure of influence a particular node has on a network based on relative scores assigned to each individual node and are scored higher if they have more connections (edges). In our network, the most influential node would be the actor(s) who appeared in the most films. When we evaluate the centrality of the nodes in our graph, we hope to find out who the most dominant actors are in Hollywood. Finding this out will also help us see if acting in more movies equates to more success.

1.2 Applications

Starting off with link prediction, we can see a few impactful applications in the real world. A predicted link in our network would represent the likelihood that two actors will work together on a future film. This information could help Hollywood directors/executives with casting the right pair of actors for a specific type of movie. Some potential applications using community analysis includes helping filmmakers see who a certain actor acted alongside, or who are some potential actors, that are in the same community, that makes sense to cast alongside. For example, Al Pacino is a great actor when it comes to mafia style movies, so if another actor who hasn't worked with Al Pacino but is also a great actor in mafia movies (so they are in the same community), then they would be a great pair for a future mafia related movie. The major application for centrality would be that given an actor, find out how prominent they are in the Hollywood film industry. This information would give actors an estimated look at their worth so they can negotiate their salary more fairly, which is a huge concern especially for female actors in today's film industry.

2 PROBLEM DEFINITION

Our network can be described as follows, given a graph $G = (N, E)$ with the set of nodes $|N|$ and edges $|E|$ the actors will be represented as nodes with an edge E between two nodes N and N' if the two actors have appeared in the same movie.

2.1 Community

Community detection, more formally, is defined as finding subsets of nodes, $(S \subset N)$, within a graph such that the connections within the subset of nodes are more dense than the rest of the network. When we talk about the graph density, we simply refer to the ratio of the number of edges with respect to the maximum possible edges. So for our network, since we have an undirected graph, we can define graph density as follows:

$$D = \frac{|E|}{\binom{|V|}{2}} = \frac{2|E|}{|V|(|V|-1)} \tag{1}$$

where $|E|$ is the number of edges and $|V|$ is the number of vertices in the graph. The maximum number of edges for an undirected graph is

$$\binom{|V|}{2} = \frac{|V|(|V|-1)}{2} \tag{2}$$

so the maximum density is 1 and the minimum density is 0 [6].

2.2 Centrality

Node centrality is a measure of influence/importance a particular node N has on a graph, $G = (N, E)$, based on relative scores assigned to each node. The scores can be determined in a variety of ways; and so centrality depends on what we choose to consider important.

2.3 Link Prediction

Given a graph $G = (N, E)$ which has a set of edges $|E|$ at time T_1, we want to predict the set of new edges $|E1|$ that are the most likely to form, such that $|E1| \not\subset |E|$, by time T_2 such that $T_2 > T_1$. Namely we are predicting edges that are most likely to be formed in the future.

3 RELATED WORK

Detecting the set of the most influential actors is similar to the key player problem and has been a focus of many studies [10]. A number of projects are based on solving this problem by using betweenness and centrality. We used the paper on Hollywood Community Detection by Barton, Manoj and Wang as a base for our implementation to detect communities within the network. They have based their analysis on two models:

(1) Baseline: This model uses the Louvain Algorithm and the Clauset-Newman-Moore community detection algorithm. Similar to this, we have used the Girvan-Newman and Louvain Algorithm.
(2) Graph Neural Networks (GNN): They applied a deep learning approach to improve the performance. GNNs can improve performance by aggregating information from the neighbouring nodes which can then be used for optimization of the communities. The target is to produce a node label based on the state of the nodes (edges, strength, and other features). Determining the state is an iterative process. The authors also talk about graph convolution networks. This approach can be useful in solving the problem of overfitting on local neighbourhood structures. To further improve the quality of their analysis, they also used a graph attention network (GAT) which adds an attention coefficient to node features. This stabilizes the learning process for the network.

Instead of studying the different variations in GNNs, our implementation focuses on using weighted edges to determine how strongly connected two nodes (actors) are related to each other which can give us a different perspective of the results. While their sole focus is community detection, we tend to focus on a more diverse set of applications which uses these weighted edges for link prediction and centrality too. A key difference in our implementation is that we are analysing the data based on the movies released during a particular period of time. Their dataset shows more centrality for the actors who have already finished their career. As a result of this, those actors will be more prominent than the more recent ones.

4 METHODOLOGY

4.1 Community

There are several algorithms that we used for community detection, this was done in order to verify if similar communities would be detected.

4.1.1 ***Girvan-Newman Algorithm.*** The Girvan-Newman algorithm is a community detection algorithm, for undirected unweighted networks, that depends upon the iterative elimination of edges with the greatest edge betweenness, where edge betweenness refers to the number of shortest paths passing over an edge [8]. We can describe the Girvan-Newman algorithm in the following way [15]:

(1) Repeat (loop) until no edges are left:
   (a) Calculate the edge betweenness for every edge in the graph
   (b) Remove the edges with the highest edge betweenness
(2) The connected components left are the communities

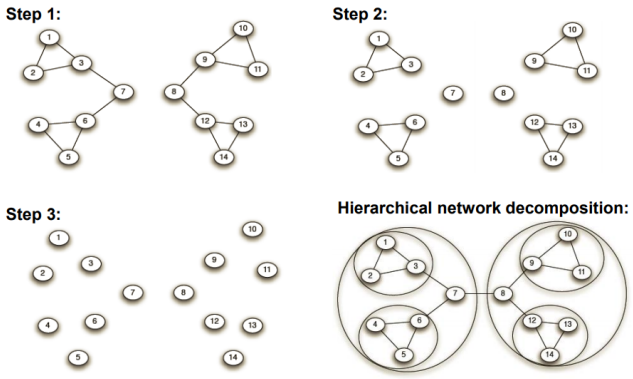

Figure 1: The steps used in the Girvan-Newman algorithm

The running time of the Girvan-Newman algorithm is

$$O(m^2 n) \tag{3}$$

where m is the number of edges and n is the number of nodes. When removing the edges with the highest betweenness value, we get the complexity $O(m)$ since there are m edges we have to remove. For each of these edges, we have to perform the edge betweenness which takes $O(mn)$ time, so overall, the complexity becomes $O(mn * m) = O(m^2 n)$.

For the Girvan-Newman algorithm, the main mathematics is needed when computing the edge betweenness. In order to calculate

the edge betweenness, we need to find all the shortest paths in the graph. The Girvan-Newman algorithm starts with one node, say A, and then performs a Breadth First Search starting from that node to create a tree. We then count the number of shortest paths from A to all other nodes in the network, so at the end each node will have a number corresponding to the number of shortest paths from A. Counting the number of shortest paths can be shown through the following algorithm [12]:

```
while L is not empty:
    Pop i ← L
    For each vertex j ∈ Adj(v)
        If d_i < d_j then b_i = 1 + Σ_j^∞ α_ij
        If d_i > d_j then α_ij = (w_j / w_i) * b_i
```

Figure 2: Counting number of shortest paths using Girvan-Newman

While this algorithm is simple to understand, it does have some limitations. The algorithm isn't very efficient with networks containing a large number of nodes and data since the communities in huge and complex networks are difficult to detect [14]. There were various variations of this algorithm which used the idea of "Modularity", namely the Claust-Newman-Moore algorithm, which had a much better complexity of $O(mdlog(n))$ but since it optimises modularity, smaller clusters aren't detected in large networks. Smaller clusters aren't detected because the expected number of edges between two clusters gets smaller as the network size increases therefore if random edges are formed between two clusters the algorithm will merge the two clusters into one large cluster and since our network is large it was not a viable option [13].

4.1.2 ***Louvain Algorithm.*** The Louvain Algorithm is another community detection algorithm that aims to maximize the modularity score of each community and evaluate how much more densely connected a community of nodes is compared to a random network [4]. It can be broken down into two parts: modularity optimization and community aggregation. During modularity optimization, nodes are greedily assigned to communities such that local modularity is optimized. During community aggregation all nodes of the same community are merged into one giant node, links connecting the new giant nodes are based on the previous graph's edges that connected the nodes between communities.

We can describe the Louvain Algorithm in the following way:

(1) Repeat until stage 1 produces no more reassignments:
  (a) Stage 1: Modularity Optimization
    (i) Repeat until one walk through of no nodes being reassigned to a different community.
      (A) Iterate through each of the nodes in the network: for each node consider the change in modularity if the node was removed from its current community and placed into a neighbouring community as well as the change in modularity for the neighbouring nodes. If the overall changes are positive the node is reassigned to the new community otherwise it remains unchanged.
  (b) Stage 2: Community Aggregation
    (i) The communities found in stage one are used to form a new network where all the nodes of the same community are merged into one giant node. The edges between two communities is just the sum of edge's weight from the original network and the linking within each community are now self loops for each of the new giant nodes.

This algorithm has a running time of

$$O(nlog^2 n) \tag{4}$$

and this is due to the fact that during stage 1 we iterate over n nodes and since this algorithm is repeated until no more reassignments of communities are produced, this change is observed to require $log^2 n$, thus getting us the complexity of $O(nlog^2 n)$.

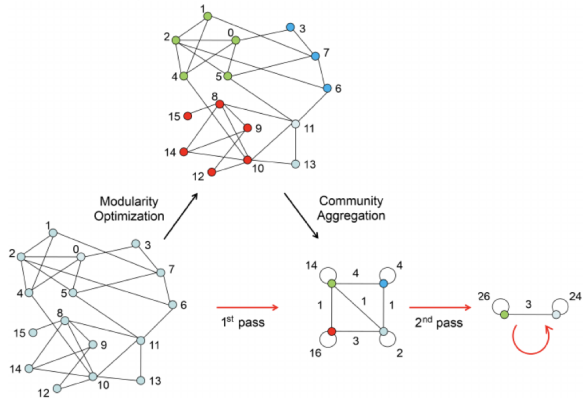

Figure 3: The sequence of steps followed by the Louvain Algorithm

4.2 Centrality

For analyzing node centrality we used three different algorithms, each algorithm has a different way of accessing the "importance" of nodes and so it will be important to compare the results of each algorithm against each other.

4.2.1 ***Degree Centrality.*** For Degree Centrality analysis, each node is assigned a value of 'centrality' based purely upon its degree. The higher number of edges a node has, the more central it is. It is a naive approach which takes advantage of the fact that many nodes with high degree are also considered highly central by various other algorithms. However, the quality of each connection is not considered, and this can result in a poor measurement of centrality. Other algorithms such as Eigenvector and Harmonic centrality will account for the centrality of all connected nodes when assigning a value to a given node. In order to find the degree centrality, the runtime complexity is $O(n^2 m)$, where n is the number of nodes and m is the number of edges [3].

4.2.2 ***Eigenvector Centrality.*** The eigenvector centrality algorithm determines the centrality of each node based on the centrality of its neighbouring nodes, meaning a node is more important if it has more important neighbours [7]. Scores are assigned to each node such that higher-scoring neighbours, N_1, add more to a node's score than an equal number of connected to lower scoring nodes, N_2. The eigenvector calculation is done by the power iteration method which has no guarantee of convergence but does have a

max iteration of n^3 and thus this algorithm has a complexity of $O(n^3)$. Below is the pseudo code for this algorithm [1]:

```
# Eigenvector Centrality (direct method)
# INPUT: g = graph, t = precision
# OUTPUT: vector = centrality vector,
# value = eigenvalue,
# iter = number of iterations

eigenvector.centrality = function(g, t) {
  A = get.adjacency(g);
  n = vcount(g);
  x0 = rep(0, n);
  x1 = rep(1/n, n);
  eps = 1/10^t;
  iter = 0;
  while (sum(abs(x0 - x1)) > eps) {
    x0 = x1;
    x1 = as.vector(x1 %*% A);
    m = x1[which.max(abs(x1))];
    x1 = x1 / m;
    iter = iter + 1;
  }
  return(list(vector = x1, value = m,
         iter = iter))
}
```

Figure 4: The pseudo code for the Eigenvector Centrality algorithm

4.2.3 ***Harmonic Centrality.*** The Harmonic degree centrality algorithm computes the Harmonic Centrality for each node [9]. The Harmonic centrality of a node u is the sum of the reciprocal of the shortest path differences from all other nodes to u, and the formula is as follows:

$$C(u) = \sum_{v \neq u} \frac{1}{d(v,u)} \tag{5}$$

where $d(v,u)$ is the shortest-path distance between v and u. If no such path exists between v and u, then

$$\frac{1}{d(v,u)} = 0 \tag{6}$$

Higher values indicate a higher centrality.

The computational complexity of the Harmonic Centrality algorithm is

$$O(nm) \tag{7}$$

where n is the number of nodes and m is the number of edges. This runtime is the same as the Closeness Centrality algorithm.

4.3 Link Prediction

For link prediction, we found three different algorithms to use to predict links and compared each of the algorithm's results.

4.3.1 ***Preferential Attachment Algorithm.*** The preferential attachment model operates on the idea that if a node is more connected, then it will most likely receive more incoming edges making it ideal for social networks [2]. Hollywood often operates like a social network, where actors that tend to do a lot of blockbuster movies in the past also tend to be the actors that will get roles in blockbuster movies in the future; making the preferential attachment an excellent candidate for link prediction. This algorithm computes the preferential attachment score for each node using the following formula:

$$PA(x,y) = |N(x)| * |N(y)| \tag{8}$$

where $N(a)$ is the set of nodes adjacent to a, and higher values mean two nodes are closer, while a value of 0 means two nodes are not close. This algorithm has a running time of $O(n^2)$ because it iterates over all nodes and for each node N it iterates over node's neighbours N'.

4.3.2 ***Adamic/Adar Algorithm.*** The Adamic/Adar link prediction algorithm is an algorithm that is used to predict links in a social network. As we stated in the preferential attachment algorithm, the Hollywood film industry works as a social network which makes this algorithm perfect for our network. It works by assigning large weights to common neighbors w of u and v which themselves have few neighbors (weight rare features more heavily)[5]. The Adamic/Adar index of u and v is defined as:

$$A(u,v) = \sum_{w \in \Gamma(u) \cap \Gamma(v)} \frac{1}{log|\Gamma(w)|} \tag{9}$$

where $\Gamma(u)$ denotes the set of neighbors of u. This index leads to zero-division for nodes only connected via self-loops. It is intended to be used when no self-loops are present. This algorithm is meant for undirected graphs, since it is computed using the shared neighbors of two vertices from the graph. The implementation computes the Adamic/Adar index for every pair of vertices connected by an edge and associates it with that edge. The time complexity of this algorithm is $O(E)$, where E is the number of edges, with a space complexity of $O(E)$.

4.3.3 ***Jaccard Coefficient.*** The Jaccard Coefficient algorithm is also used for link prediction. This algorithm produces a value that scores the similarity of each pair of nodes; the higher the score the higher the likelihood of the two nodes connecting in the future[11]. The score for two nodes x and y is obtained by computing the length of the intersection of the neighborhoods of x and y divided by the length of the union of the neighbourhoods of x and y; formally defined as the following formula:

$$score(x,y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x) \cup \Gamma(y)|} \tag{10}$$

This shows how similar two nodes' circle of trust is and also factors in the size of each node's neighbourhood so that nodes with larger neighbourhoods don't have an unreasonable advantage; i.e. filtering out actors that just do small roles in a lot of movies. Furthermore, this methodology should translate very well to a Hollywood network because A-list actors tend to do more roles with other A-list/B-list actors and so on; meaning in theory the closer two actors' are in terms of their circle of trust the more likely they are to do movies together in the future; especially since the film industry is often about who you know. This algorithm can be implemented in two ways:

Option A: Neighbourhoods for each node are computed and sorted based on node id. This is done by iterating through each

neighbourhood simultaneously checking for collisions (iterate only through nodes with a lower assigned id). Once all the intersections are computed the union can be computed in constant time since the size of each neighbourhood is known. This algorithm runs in $\theta(Mlog(M))$ where M is the size of the largest neighbourhood.

Option B: The brute force method is to compare each neighbourhood pair and it has a running time of $\theta(M^2)$ where M is the size of the largest neighbourhood.

5 EVALUATION

In order to determine the success/failure of our network analysis (node centrality, link prediction, and community detection) we will discuss the data sets used to generate each network analysis, initial finds for each analysis and experiments performed to validate the analysis.

5.1 Filtering

Before running analysis for centrality, three separate data sets were created from the original data set by filtering based upon parameters. Data was filtered on two parameters; movies were filtered by their release date and actors were filtered based upon how many movies each actor has acted in. The number of years of movies to include in the data set is given by t and the number of movies that each actor has acted in is given by s. The following parameter sets, in the form (t, s), were used to create three different data sets: [(40,10), (60, 20), (80, 20)].

5.2 Centrality

For centrality, three different algorithms were selected and each algorithm runs once on each new data set generated in 5.1.

5.2.1 ***Results***. The results for movies from 1977 - 2017 and actors who have acted in more than 10 movies are displayed in tabular form below:

Degree Centrality	Eigenvector Centrality	Harmonic Centrality
Robert De Niro	Robert De Niro	Robert De Niro
Val Kilmer	Val Kilmer	Val Kilmer
Jon Voight	Jon Voight	Jon Voight
Tom Sizemore	Tom Sizemore	Tom Sizemore
Diane Venora	Diane Venora	Diane Venora
Ashley Judd	Ashley Judd	Ashley Judd
Natalie Portman	Natalie Portman	Natalie Portman
Ted Levine	Ted Levine	Ted Levine
Tom Noonan	Tom Noonan	Tom Noonan
Al Pacino	Al Pacino	Al Pacino

The results for movies from 1957 - 2017 and actors who have acted in more than 20 movies are displayed in tabular form below:

Degree Centrality	Eigenvector Centrality	Harmonic Centrality
Anthony Hopkins	Anthony Hopkins	Anthony Hopkins
Joan Allen	Joan Allen	Joan Allen
Ed Harris	Ed Harris	Ed Harris
E G Marshall	E G Marshall	E G Marshall
David Hyde Pierce	David Hyde Pierce	David Hyde Pierce
Paul Sorvino	Paul Sorvino	Paul Sorvino
Mary Steenburgen	Mary Steenburgen	Mary Steenburgen
J T Walsh	J T Walsh	J T Walsh
James Woods	James Woods	James Woods
Kevin Dunn	Kevin Dunn	Kevin Dunn

The results for movies from 1937 - 2017 and actors who have acted in more than 20 movies are displayed in tabular form below:

Degree Centrality	Eigenvector Centrality	Harmonic Centrality
Anthony Hopkins	Anthony Hopkins	Anthony Hopkins
Joan Allen	Joan Allen	Joan Allen
Ed Harris	Ed Harris	Ed Harris
E G Marshall	E G Marshall	E G Marshall
David Hyde Pierce	David Hyde Pierce	David Hyde Pierce
Paul Sorvino	Paul Sorvino	Paul Sorvino
Mary Steenburgen	Mary Steenburgen	Mary Steenburgen
J T Walsh	J T Walsh	J T Walsh
James Woods	James Woods	James Woods
Kevin Dunn	Kevin Dunn	Kevin Dunn

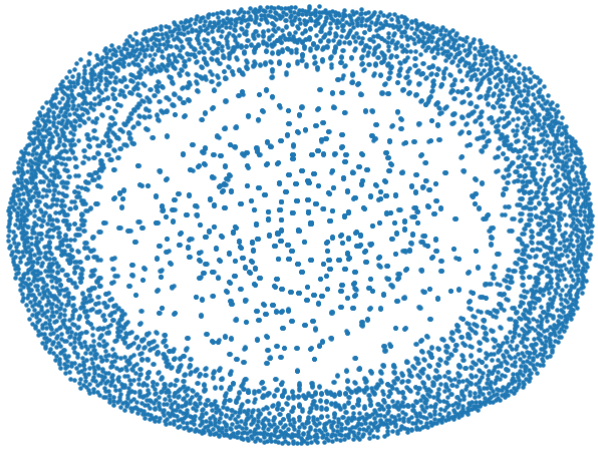

Figure 5: The network generated for Node Centrality using the Degree Centrality algorithm

5.2.2 ***Discussion***. Notably, the results varied based upon what filtering parameters were used and this is as was expected. When the data was filtered to include the past 60 to 80 years of movies, the names which appear are actors with an expansive filmography, who were stars in the mid 1900s. When filtering the data set to the past 40 years of movies, the names that appear are much more familiar and are recognizable Hollywood actors who are popular and are well known today. The most influential nodes in the past 60 and 80 years of acting do not significantly differ outside of the top 10, implying that the determined top 10 nodes are correctly the

most central for that time period. Given that an actor's career can last more than 20 years, this seems reasonable.

Interestingly, the results did not vary when different algorithms were used. For each data set, the result for most central actors was identical when calculated using Degree, Harmonic, or Eigenvector centrality. The only information that can be used for each algorithm would be related to the degree of each node and this is believed to be the cause for this output.

5.3 Link Prediction

The dataset selected from Kaggle provides all the movies spanning between 1874-2017, but for link prediction our dataset used the set of movies from 1960-2000, i.e. edges formed until 2000. To begin with, we found the set of friends of friends, or pairs of nodes that had a shortest path length of 2, which became our candidate edges. We used the link prediction algorithms in order to find the preferential attachment score, Jaccard Coefficient and the Adamic-Adar score for each candidate edge. The candidate edges were sorted based upon their scores and the top k scores were selected to calculate precision. Precision @ k was found for k values of 10, 20, 50, and 100. As the value of k increased, the Precision @ k score decreased, which makes sense. The value of k represents the top k pairs of nodes which the algorithms valued. This means that as we increase the value of k, we are including pairs with a lower score or coefficient assigned to them. As a result, we would expect that the precision would generally decrease because we are including more inaccurate predictions. Tables for each algorithm's output, at $k = 10$ are displayed in tabular form in **5.3.1**.

5.3.1 ***Results***. The table for the pair of actors for each of the top 10 Preferential Attachment scores are:

Actor 1	Actor 2	Preferential Score
John Travolta	Al Pacino	30810
Michael Ironside	John Travolta	29640
Lance Henriksen	John Travolta	28275
John Travolta	Walter Matthau	28080
Nicole Bilderback	John Travolta	27690
George Clooney	John Travolta	26910
Ariane Schluter	John Travolta	26325
Mia Farrow	John Travolta	26325
Jon Tenney	Al Pacino	26228
George Clooney	Al Pacino	26191

The table for the pair of actors for each of the top 10 Adamic-Adar scores are:

Actor 1	Actor 2	Adamic-Adar Score
Daryl Gates	Ricky Harris	12.053
Jack Nicholson	Lance Henriksen	11.509
Nicole Bilderback	John Travolta	11.285
John Travolta	Mia Farrow	11.203
Stephen Boxer	Lance Henriksen	11.191
Marisa Tomei	Helen McCrory	10.842
Jon Tenney	Gary Oldman	10.230
Jack Nicholson	Stephen Boxer	10.139
Jon Tenney	James Garner	10.123
Jon Tenney	Al Pacino	10.001

The table for the pair of actors for each of the top 10 Jaccard Coefficient scores are:

Actor 1	Actor 2	Jaccard Coefficient
Patrick Chesnais	Marc Maurette	1.00
Eli Marienthal	Tony Walker	1.00
Jason Mewes	Michael Rooker	1.00
Tony Walker	Eli Marienthal	1.00
Adrian Paul	Kelly Lin	0.60
Emil Chow	Kelly Lin	0.60
Karan Dewani	Oh Ji hye	0.50
Kajol	Vidya Balan	0.50
Justin Kirk	Raiz Ichikawa	0.50
Kelly Lin	Maria Doyle Kennedy	0.44

The Precision @ k scores when $k = 10$ for Preferential Attachment, Adamic-Adar and Jaccard coefficient were 0.4, 0.35 and 0.28, respectively.

5.4 Communities

The result of running the Louvain Algorithm on the raw data set was that approximately 45,000 communities were detected. Upon closer inspection of the output, it was determined that each community detected represented a movie. This lines up with the number of movies included in the data set. To explain this, a deeper analysis of the network's definition is required. When the network was generated, as defined in Section 1, actors were treated as nodes and when two actors acted in the same movie, an edge was created between them. This means that it is possible to treat every movie as a complete sub graph with $n * (n-1)/2$ edges and n nodes. This resulted in extremely dense, complete subgraphs with only a few major actors branching out - the algorithm correctly grouped these actors together. This was verified by comparing the outputted communities to the raw data set and figuring out that each community consisted of only the actors in each movie.

Instead of using the Louvain Algorithm, the Girvan-Newman algorithm was used. The Girvan-Newman algorithm works by iteratively removing the edges which are considered to be the most 'between' for all communities. An edge is always removed at each iteration, so the number of communities formed is decided by the group.

It was decided that a value of 10 was to be used for the number of iterations, resulting in 10 communities being formed. For the first few communities generated, their size was simply one, meaning that an extremely isolated actor who did not belong to any other community broke off from a community in the previous iteration to form a lone community. By around iteration 6, communities were beginning to be formed which the group noticed had names of actors who worked in multiple film industries, such as both Bollywood Hollywood or both the Japanese Film industry Hollywood. This was no doubt due to the fact that individuals who work in an industry outside of Hollywood would be more likely to work each other in a Hollywood context as well. The communities formed are displayed in tabular form, where C represents the community number:

C	Actors
1	{Joel Stedman}
2	{Ilya Lagutenko}
3	{J. Scott Smart}
4	{Evelyne Kraft}
5	{Chandan Roy Sanyal}
6	{Raiz Ichikawa, Daisuke Ry}
7	{Sumiko Takai, Manami Fuji, Keichi Ueda, ... }
8	{Sergey Gazarov}
9	{Kajol, Shah Rukh Khan, Pradeep Ram Singh Rawat, ... }
10	{Whitney Houston, Al Pacino, Val Kilmer, ... }

Due to space constraints, only the first three members are listed in each community. The true sizes of communities 7, 9, and 10 are 5, 13, and 3383, respectively. The full results for the community detection algorithms can be found at the GitHub link (found after the conclusion) under the 'community' subfolder.

6 CONCLUSION

We processed the data and constructed the network from scratch. After parsing the data from the files, our initial data set consisted of almost 200,000 nodes and 1,500,000 edges. We cleaned our network on the basis of the number of movies the actors have been a part of and also by specifying the period of time we want to focus on. We focused primarily on three pairs of time period and number of movies for each actor. These times periods can be shown in the table below:

#Years	#Movies	#Nodes	#Edges
40	10	8897	9636
60	20	3407	3511
80	20	3450	5784

The centrality analysis for our network using Degree, Eigenvector and Harmonic Centrality algorithms were identical and different results were obtained by filtering the datasets with different parameters as expected.

Link prediction worked well, but the dataset considered covered a time period of 40 years while the precision @ k was found using the next 20 years of film industry. Due to this, there was no doubt some aspect of random chance of two actors working together anyways due to the law of large numbers and the scale of time being considered.

The Louvain Algorithm detected almost 45,000 communities which was expected because the movies themselves represent a very strongly connected dense network. The Girvan-Newman method was used instead to partition the network into precisely 10 communities to obtain a more meaningful result. What was noted was that actors were grouped based upon industries they had worked in outside of Hollywood. For example, Bollywood actors who had also worked in Hollywood ended up forming communities in Hollywood as well.

6.1 Future Work

In the future, we intend to add a more extensive set of parameters in order to perform more accurate link prediction and centrality analysis. We would also aim to determine algorithmic results for different sets of parameters such as graphs cleaned with different methods and also perform community detection or link prediction on a smaller, year by year scale.

Using a different layout for the structure of the network, such as a different representation of edges rather than acting in movies together, or adding weights to the current edges based upon user reviews have been considered by the group. Given another opportunity to work on the project or further extend it, the group would consider a common film between two actors, the film's financial success and use this value to indicate a successful partnership. It is the group's belief that this would result in more accurate centrality measures than simply degree based measures. Link prediction algorithms which consider the quality of a link could also be used, and this would then result in predictions which are fiscally beneficial for Hollywood casting agencies, allowing practical application of the somewhat theoretical work done in the project.

The group would also consider using a bigger dataset in order to verify and work on the discovery of communities being formed between international actors in Hollywood. Using a more comprehensive or international dataset may result in different communities being formed and may allow the group to use the Louvain Algorithm to determine a different set of communities.

GITHUB

This is the link to our public GitHub repository:
https://github.com/arneshie/movie_star_network/

REFERENCES

[1] [n.d.]. https://www.sci.unich.it/~francesc/teaching/network/eigenvector.html
[2] [n.d.]. https://neo4j.com/docs/graph-algorithms/current/labs-algorithms/preferential-attachment/
[3] [n.d.]. Centrality Measures For Networks.
[4] [n.d.]. Community detection for NetworkX's documentation¶. https://python-louvain.readthedocs.io/en/latest/
[5] [n.d.]. Community detection for NetworkX's documentation¶. https://python-louvain.readthedocs.io/en/latest/
[6] 2020. Dense graph. https://en.wikipedia.org/wiki/Dense_graph
[7] 2020. Eigvenvector Centrality. https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.centrality.eigenvector_centrality.html
[8] 2020. Girvan-Newman. https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.community.centrality.girvan_newman.html
[9] 2020. Harmonic Centrality. https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.centrality.harmonic_centrality.html
[10] Sophia Barton, Nidhi Manoj, and Flora Wang. 2019. Hollywood Actors Community Detection And Genre Prediction. (2019).
[11] Neil Butcher. [n.d.]. Jaccard Coefficients.
[12] Ljiljana Despalatović, Tanja Vojković, and Damir Vukicevic. 2014. Community structure in networks: Girvan-Newman algorithm improvement. In *2014 37th international convention on information and communication technology, electronics and microelectronics (MIPRO)*. IEEE, 997–1002.
[13] Ying Ding. 2011. Community detection: Topological vs. topical. *Journal of Informetrics* 5, 4 (2011), 498–514.
[14] Mehak Mohammad. 2018. "Community Detection in Social Networks Through Girvan Newman Algorithm". https://medium.com/@96mehakmohammad/community-detection-in-social-networks-through-girvan-newman-algorithm-78f303912907
[15] Luís Rita. 2020. Louvain Algorithm. https://towardsdatascience.com/louvain-algorithm-93fde589f58c