Graph and Semantic Analysis of Movie database

Sara Nikula
University of Oulu, Degree Programme in Computer Science and Engineering
Oulu, Finland
sara.nikula@student.oulu.fi

Timo Mattila
University of Oulu, Degree Programme in Computer Science and Engineering
Oulu, Finland
timo.mattila@student.oulu.fi

***Abstract* — This paper is a project work report for the course Social Network Analysis, Spring 2020. We have created and studied graphs based on a dataset which contains information about over 5000 movies and their actors. We created two graphs, one consisting of actors as nodes and two nodes being connected via an edge if the two actors had played in at least one common movie. The other graph consisted of movies as nodes and any two nodes were connected via an edge if there was at least one actor who had played in both of the movies. We examine communities that can be found in these graphs, centrality measures of individual nodes and histograms of centrality measure distributions. We also try if it is possible create a random graph similar to the original graph by varying probabilities of edge or triangle creation.**

Keywords — movies, graph, social network analysis, cliques

I GROUP INFORMATION

Our group is group number 7 and we worked on this project as a pair. The title of our project is *Graph and Semantic Analysis of Movie Database*.

II INTRODUCTION

Internet movie database (IMDb) is an online database of movies, tv shows, video games, streaming content and more. It contains information on the production, casting, actors, trivia, and allows the users to rate and review the content [1]. IMDb was first launched in 1990 on Usenet and was moved to web in 1993. It was acquired in 1998 by Amazon.
As of 2019, IMDb has records of over 538,000 movies and 120,000 TV movies [2]. Therefore, the IMDB 5000 dataset represents 0.91% of the movies in the database. IMDb is mainly volunteer based, where users add content to the site for free. IMDb also provides pro accounts for 20$ per month, where the user gains access to more features. IMDb pro accounts are available for movie industry professionals for free.

III PROBLEM DESCRIPTION

We were tasked to create graphs of over 5000 movies and their actors and analyze the graph in various ways. Our goal was to find out how actors are linked together via movies, how they form communities and who are the bridges between these communities. We did not have to collect the database on ourselves. We were given a ready-made open dataset to work with.

IV DATASET DESCRIPTION

Our dataset was IMDb 5000 movie database which has information about 5000 movies and actors who have played in them [1]. The original version of the dataset was created on Kaggle, but it was taken down due to a DMCA takedown request by IMDB

[3]. Thus, it has not been updated since, which is reflected in numerous errors in the database. A new version has since been created, which is different from the one used in this study.

Each movie row contains the following fields (reordered for clarity, the actual data has different order and names for columns):

1. Title of the movie
2. Director
3. Director Facebook likes
4. First actor
5. First actor Facebook likes
6. Second actor
7. Second actor Facebook likes
8. Third actor
9. Third actor Facebook likes
10. Total Facebook likes of the cast
11. Movie Facebook likes
12. Number of human faces in the movie poster
13. Number of reviews by critics
14. Number of reviews by users
15. Number of user votes
16. IMDB score
17. Budget, in dollars or local currency
18. Gross, in dollars or local currency
19. Genres
20. Plot keywords
21. Color of the movie
22. Duration, in minutes
23. Aspect ratio
24. Release year
25. Content rating
26. Country
27. Language
28. IMDB link

Dataset was cleaned before further analysis. If a movie appeared multiple times in the dataset, the extra copies were skipped. Movie titles with commas and semicolons (e.g. *"It's a Mad, Mad, Mad, Mad Word"* and *"Sanctuary; Quite a Conundrum"*) caused problems during parsing the dataset. Initial tries to consume the data resulted in cases where the data got jumbled during reading, as the rows were split into too many fields. This caused data to be read in the wrong field, which resulted in cases where e.g. some actor names were replaced with the budget of the movie. Additional problems were caused during the initial review of the dataset with a spreadsheet program, as the program turned some of the fields into different format. For example, the IMDB rating and aspect ratio were read as dates and transformed into day-month format. We recommend later studies to be careful when examining the file, and downloading the dataset from its original source, in order to make sure the data stays the correct. The dataset is in **UTF-8** format, opening it in other formats might cause problems when displaying movie titles and actor names which contain letters outside the English alphabet.

The dataset also contained mistakes in the data itself. These mistakes fell into two categories:

1. Wrong movie
2. Erroneous data in columns.

The first category refers to cases where the movie row does not contain the suspected right movie. For example, the title of the movie might refer to some big budget film, but the actual data and the IMDB link to the movie entry are for another movie, or for a *video review* of the actual movie. The second category of mistakes refer to mistakes in parts of the movie data. Both of these two categories of mistakes are hard to identify and correct. Therefore, they have been left in place, and some noise should be assumed to be in the end results.

It was noted that the first, second and third actors were all actors who played a role in the film, but were not necessary part of the main cast. In more than one entry it was observed that the leading actors were not present in the three actors listed for the movie. For example, some of the *James Bond* movies did not contain the actor of James Bond. This should not be assumed to be a mistake. Instead, it should be assumed that the actors in these movies were chosen randomly for the dataset. This has been confirmed by the creators of the dataset on Kaggle , who have said that the database does not contain actors in any identifiable order [3], although there was some discussion on the Kaggle site that the actors could have been chosen based on their Facebook likes. The terms first, second and third actor should not be mixed with typical movie terminology, where actors with largest role in the movie are often called either the *leading actor*, *starring role* or *title character* [4]. Therefore, the set of actors in the data set will be larger than the set of actors if the first, second and third actor were the three highest billed actors in each movie.

V General methodology

Our general methodology is based on building and examining the graphs via programs written on Python. We utilize NetworkX package, which is especially suitable for creating, manipulating and analyzing complex networks [5]. NetworkX provides several ready-made functions for analyzing the most central aspects of a graph [6]. First, we created a graph from our dataset and after that utilized NetworkX functions and some self-written functions to analyze properties of these graphs.

VI Detailed methodology

In this section we discuss more detailed the methodology which we used to solve the various tasks. Each of the subsections correspond to one of the ten tasks we were given.

6.1 Building graph and basic graph measures

In task 1 we examined the first graph, which consisted of actors as nodes and an edge between any two nodes if the actors had played in at least one common movie. We created the graph with NetworkX by going through all movies in our dataset and adding a new node to our graph whenever a new actor was mentioned. An edge was added between all actors mentioned in connection of a movie. We then studied general properties and key characteristics of this graph: number of nodes and edges, overall clustering coefficient, diameter, size of the giant component, average path length, number and quality of communities. In addition to these four different centrality measures were examined: in-betweenness centrality, eigen vector centrality and degree centrality.
We calculated these measures by using ready-made functions of NetworkX: degree(), average_clustering(), diameter(), average_shortest_path_length(), eigenvector_centrality() and betweenness_centrality(). We used also adjacency_matrix() in order to display the first 8 x 8 elements of adjacency matrix of this graph. The graph was not fully connected but included one giant component containing over 2/3 of the nodes. It was not possible to calculate average shortest path or diameter for a disconnected graph, because distance between nodes in separate components is infinite. Instead we calculated these measures for the giant component only, using NetworkX function connected_components() to extract the giant component.
In order to detect the number of communities we first tried Girvan-Newman implementation of NetworkX. This turned out to be hopeless as this algorithm has time complexity O(m^2n) [7] so that utilizing it to calculate communities of our graph would take too long. Instead of Girvan-Newman we used ready-made function best_partition() [8] and examined the quality of this partition by using functions modularity(), coverage(), inter_community_non_edges() and performance(). These functions belong to the community module of NetworkX [9].

6.2 Degree distribution of actors

In task 2 we had to investigate the degree of distribution of the graph. Calculating this was simple, we simply counted how many degrees each node had and calculated what fraction of nodes had a degree on one and so on.
We were also tasked to study whether the distribution follows the power-law distribution. We investigated this by using *scipy.optimize.curve_fit* function with custom *power_law* function. We only tried to fit the even-numbered degrees, because they seemed to follow possibly follow the distribution.

6.3 Top 10 actors based on Centrality measures

We investigated top 10 actors ranked according to different centrality measures: degree centrality, eigenvector centrality, in-betweeness centrality and closeness centrality. Implementation was quite straightforward with ready-made NetworkX functions: nx.degree() for degree centralities, nx.eigenvector_centrality() for eigenvector centralities, nx.betweenness_centrality() for in-betweenness centralities and nx.closeness_centrality() for closeness centralities. We utilized these functions with our actor graph, sorted the output lists by centrality value in descending order and collected the first 10 elements of every list.

6.4 Communities

Task 4 included studying communities in the graph using different definitions for a community. As

already mentioned in section 6.1 we could not utilize Girvan-Newman implementation of NetworkX for community detection because of its time complexity and the size of our graph. Detecting cliques in the graph was simple with ready NetworkX function find_cliques(). This function returns a list of maximal cliques found in the graph [10]. A clique is a maximal complete subgraph containing a given node, which means that all nodes in the subgraph are connected with each other. K defines how many nodes the subgraph has. To find out the amounts of different clique sizes we iterated through all cliques in the list, classified them into a dictionary according to their size (k) and examined how many cliques a given k includes.
Cores in the graph were calculated with NetworkX function k_core(). This function returns a maximal subgraph that contains only nodes of degree k or more [11] We utilized this function with different k's to detect cores of all sizes.

6.5 Investigating how communities form around genres

Task 5 included investigating the communities found in the dataset and studying if they correspond to movie genres or series. After inspecting the data we noticed that reliably identifying if two movies were belonging to the same series was challenging, because the only way to detect this was to inspect plot keywords of the movies. Even though a movie being a prequel or a sequel of another movie was usually mentioned in the description, there was no consistent convention. Extracting the desired information therefore would have required advanced automatic language processing skills which we do not possess. Therefore, we decided to only focus on genre.

For this task we constructed another graph. This graph has movies as nodes and an edge between any two nodes if the movies had at least one common actor. There were in total 4909 nodes and 44925 edges in this graph. We decided to define community here as a clique, because calculating cliques was straightforward with a ready NetworkX implementation (as explained in 6.4). A clique was also a simple and comprehensible concept for this purpose. The biggest clique size found in the movie graph was 53-clique, which is remarkably bigger than the maximum clique size in the first graph.

Our hypothesis was that a movie clique is formed on the ground of genre, so that movies belonging to the same genre would form a clique. Our further hypothesis was that genre distributions of bigger cliques would resemble genre distribution of the original graph more than those in smaller cliques. To test these hypotheses, we first extracted all cliques in the graph. Then we calculated which genre was most common in that specific clique and how many of the movies in that clique belonged to this genre. This was iterated through all cliques of all clique sizes. Our aim was not to examine which genres were most common but if there was some genre which would cover almost all the movies in that clique.
We calculated average percentages of the coverage of the dominating genre in each clique size and compared this to the overall genre distribution of the graph. Differences in genre distributions were also estimated visually by plotting genre distribution of a clique in the same figure with total genre distribution. It is worth noting that most of the movies in our dataset belonged to more than one genre.

6.6 Further community investigations

In task 6 we examined the cliques further in order to reveal if the cliques in graph 2 (movie graph) were build on ground of some other aspects than genre. We examined three different aspects: director, language and country of the movie. We used the same methodology as in task 5, modifying the code so that the instead of genre, the percentage of the dominating country, language or director was calculated. Distributions per clique were plotted together with the total graph and the results were also numerically inspected.

6.7 Actor Scoring

In task 7 we had to create a method for ranking the actors based on their movies. We set out to solve this task by laying out our requirements for the ranking system:

1. Actors are scored based on the movies they have participated in. Therefore, we need to create a method to rank the movies first.

2. Acting in multiple movies should increase the rank, but it should not dominate the score.

Movies in the database had a couple of possible indicators of their success: Facebook likes, IMDB score, different counts of reviews or votes and gross. We choose to use the IMDB score and the number of users who had voted. A naïve approach to rank the movies would have been to use the IMDB score alone, but this could lead to oddities where a movie with under 100 votes had a superb score, and thus would be high on the list. We assumed that part of the movies score must also lie in its popularity, i.e. how many have seen the movie. For ranking movies, we choose the following rules:

1. The score must be based on both the IMDB score and number of users who have voted the movie.
2. Neither of the values should dominate the scoring too much.

Based on this, we created Equation (1) for scoring movies.

$$\ln v \times 0.2 + s \times 0.8 \quad (1)$$

Where v is the number of voters, and s is the score of the movie. We take the logarithm of voters, because the range of number of votes movies had was from 5 to 1689764, with average movie having 82644 votes. Additional weights were added to balance the distribution between number of votes and IMDB scores. Despite the scoring system laying on the range from 1 to 10, most top movies were separated only by a few points (min 1.6, max 9.5, avg 6.4).

To rank the actors, we devised Equation (2).

$$\sum \frac{m_i}{max(10\times i,\ 1)} \quad (2)$$

Where m is the score of the movie. The movies of the actor are sorted in order from highest scoring movie to the lowest. Thus, every movie the actor has played in adds to their score, with diminishing returns. This fulfills both requirements. Being part in multiple high-scoring movies increases the score of the actor, and low-scoring movies do not lower their score.

For example, let's imagine we have an actor who has played in five movies with scores 100, 80, 50, 10 and 1. The actor score would be calculated like this:

100 / max(10*0, 1) + 80 / max(10*1, 1) + 50 / max(10*2, 1) + 10 / max(10*3, 1) + 1 / max(10*4, 1)

= 100 / 1 + 80 / 10 + 50 / 20 + 10 / 30 + 1 / 40

= 100 + 8 + 2.5 + 1/3 + 1/40

= 110.8583

6.8 Actor ranking and centrality measures

In task 9, we investigated how much different centrality measures correlated correlate with the actor rankings. Two centrality measures were investigated here, betweenness centrality and eigenvector centrality. We created histograms from the centrality values and actor rankings and measured the distance between these histograms using a simple bin-to-bin L1 distance (also known as city block distance, or Manhattan distance) [12], see Equation (3) for example how it was calculated. In simple words, L1 distance is measuring the difference of each column of two histograms and summing the difference together. The best possible score for L1 distance is 0, which can be achieved if two histograms are exactly the same. The maximum distance between histograms is the sum of total heights of both histograms. This can be achieved when two histograms do not overlap at all. L1 distance is a rather simplistic way to measure the distance between histograms, as it only measures each column against one column in the other histogram, i.e. bin-to-bin. More complex distance measures, like the earth-moving distance, measure the distance between bins. As our task was to measure the distance in a simple way, we opted to go with the more simplistic way of calculating the distance.

The centrality values were min-max normalized using Equation (1). This way, the results were easier to compare with the actor rankings.

$$D_1(A,\ B) = \sum_{i=0}^{b-1} \left| H_1(A) - H_2(B) \right| \quad (3)$$

6.9 Random Graph

In task 9 our aim was to construct a random graph which would as much as possible resemble the original graph. As a reference point we used our graph 1, having actors as nodes. We tested two ready-made random graph generator functions. Gnp_random_graph(n, p) creates a random graph using two input parameters: the number of nodes (n) and probability of edge creation between two random nodes (p). [13] The second function used was powerlaw_cluster_graph(n, m, p), which needs three input parameters: number of nodes (n), average degree of a single node (m) and probability of creating a triangle after creating an edge (p). [14]. The latter turned out to be better suitable for our purposes, so we iterated this function with several different parameter values and examined the resulting graph. We examined the yielded graphs by counting their average degree, average clustering coefficient, size of the giant component, diameter of the giant component and average shortest path length of the giant component and comparing these with the values of the original graph. We also inspected and estimated the histograms of our random graphs visually, comparing them to the original graph.

VII RESULTS AND DISCUSSIONS

7.1 Building graph and basic measures

We examined some general properties of our actor graph. Results are collected in the Table 1 below.

Table 1. Some general properties of the actor graph.

Number of nodes	6250
Number of edges	14326
Max degree	94
Min degree	0
Average degree	4.58
Average clustering coefficient	0.71
Giant component size	4625
Diameter	inf (not connected)
Diameter/giant component	15
Average shortest path	inf (not connected)
Average shortest path/giant component	4.88
Average eigenvector centrality	0.0045
Maximum eigenvector centrality	0.20
Maximum betweenness centrality	0.031
Mean betweenness centrality	0.00034

First 8 x 8 elements of adjacency matrix of the actor graph. Explanations for the numbers are: 0 = CCH Pounder; 1 = Joel David Moore; 2 = Wes Studi; 3 = Johnny Depp; 4 = Orlando Bloom; 5 = Jack Davenport; 6 = Christoph Waltz; 7 = Rory Kinnear.

$$\begin{array}{c|cccccccc}
 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
\hline
0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
2 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
4 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
5 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\
6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
7 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0
\end{array}$$

7.1.1 Community detection

We detected 587 communities in our graph. Modularity of this partition was 0.607, coverage was 0.651 and the performance of this partition was 0.97. It is worth noting that the partition algorithm we used is based on heuristics and thus the results are approximate. This approximate result is informative enough for our purposes though, because defining a community is not unambiguous and partitioning can be done in several ways anyway.

Modularity measures the quality of partitioning a graph into separated clusters. Modularity is maximized when the communities include many edges but there are only few edges between communities [15]. Coverage is defined by the ratio of edges inside communities and the total number of edges [16]. Note that the definitions of modularity and partitioning resemble each other, which explains the similarity in modularity and coverage indices. Considering that the graph included 587 communities and a giant component covering over

2/3 of all nodes, coverage 0.651 and modularity 0.607 are quite satisfactory. It seems clear that perfect partition of this graph cannot be done because of the size and loose composition of the giant component, which forces quite many intra-community edges.

Performance is the ratio of edges inside communities plus non-edges between communities and potential number of edges in the graph [17]. The performance of our partition, 0.97, was quite successful. One could assume that the power law character of our graph was helping a bit as there were only a few actors with particularly high degrees and a considerable amount of 3-cliques.

7.2 Degree distribution of actors

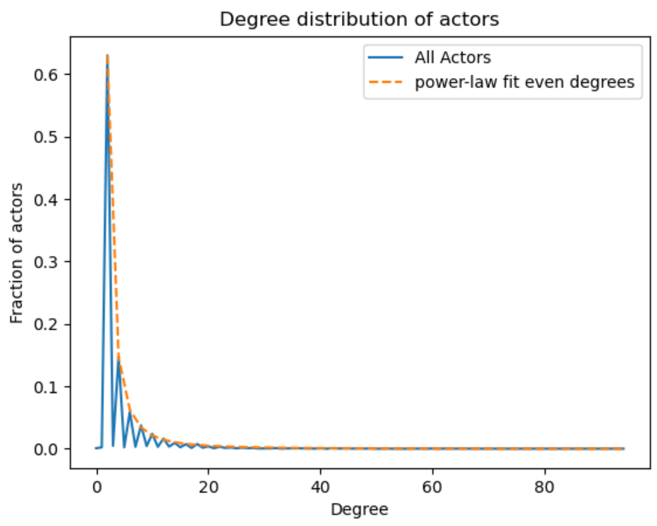


Figure 1. The Degree distribution of actors. Blue line represents the actual distribution fractions. Orange line is fitted power-law distribution on the even-degreed actors.

We tried to fit power-law distribution over the even-degreed actors, as those seemed to possibly follow the power-law distribution. The end results of this are shown in Figure 1. Equation 4 displays the power-law function which was fitted for the distribution. The fitting returned parameters *a = 2.701046±0.019570* and *b = -2.100286±0.009328.*

$$ax^b \quad (4)$$

From this we can say that the even-values seem to follow power-law distribution. However, the results are a bit sketchy due to the small sample size, as only few actors had high degree values, which introduced a lot of noise to the data. See Appendix for logarithmic version of the graph.

Interestingly, the lower range of degree distribution contains sharp spikes downwards at every odd value. 63% of actors have degree of 2, but only 0.48% have degree of 3.15% have degree of 4, but only 0.2% have degree of 5 and so on. This probably comes from the fact how each movie has three actors listed for it. Thus, every movie forms a triangle of actors, with a degree of 2, like pictured in Figure 2. Graph 1. For every movie an actor plays in, they add their degree by 2, which results in an even number, like in Graph 2. Only when two movies have two actors in common, actors with odd number of degrees appear, like in Graph 3. In addition, if two actors play together in three different films, they cancel out and end up having an even number of degrees. Movies with only two actors can generate nodes with odd number of degrees, but those movies are quite rare, as most films do have more than a pair of actors in them. This pattern of even degrees outnumbering odd degrees lasts all the way up to nodes with degree of 26, where the sheer rarity of actors who have played in over 10 movies flattens the curve.

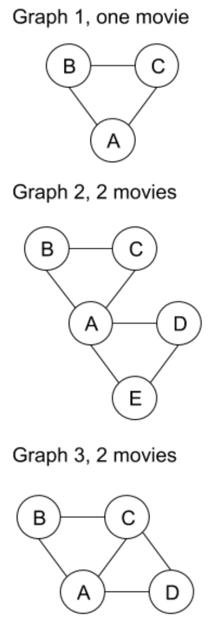


Figure 2. Example how nodes with odd and even degrees occur for actors. In Graph 2, actor A plays in two movies and thus has a degree of 4, an even degree. For odd degree to appear, two movies must have two actors in common, like in Graph 3, where A and C both play in the two movies.

7.3 Top 10 actors based on centrality

We calculated top 10 most influential persons according to four different centrality measures. Results are collected in the tables in appendix 1 – 4. Considering degree centrality, the differences in these numbers are significant in top 3 but after that they become smaller. This fits the description of power-law-distribution; only a few nodes have a remarkably high degree than the others. The average degree was 4.58. The same phenomenon can be seen in the eigenvector centralities list: eigenvector centrality of actor number one is almost twice as big as that of number 10 – there are over 6000 actors in the graph so both belong to the highest 0.17 %. The average eigenvector centrality was 0.0045. Even though the same names are dominating the top 10 again, there are also some new names which were not there in the degree centrality ranking. In the top 10 list of betweenness centralities the differences between leading nodes are even more remarkable; betweenness centrality measure of actor number 1 is more than twice as big as that of number 10. The average betweenness centrality was 0.00034. With respect to the other centrality measures, closeness centralities vary astonishingly little in the top 10. Closeness centrality of actor number 1 is only 0.01 units higher than that of actor number 10 (0.235 / 0.221).
Choice of the centrality measure has a fair impact on which nodes are considered as most central. We could conclude that the most central persons are central according to all of the four centrality measures studied, even though the top 10 list is varying a bit due to the ranking system. See for example Steve Buscemi who has the 5. biggest degree centrality but is not even mentioned in the top 10 eigenvector centralities. According to these findings the different centrality measures seem to be interconnected but not linearly dependent in this dataset.

7.4 Communities

In task 4 we detected different communities in our actor graph. As already mentioned in section 6.1 we were not able to use Girvan-Newman algorithm for community detection because of its time complexity. Clique and core detecting algorithms are ready implemented in Networkx so we used them to detect cliques and cores with different k‘s. The biggest clique size found in our dataset was 5-clique, of which there was 3 occasions in the graph. Clique amounts found in the actor graph can be seen in the list below. Note that the find_cliques() function of Networkx returns maximal cliques [10], which explains why there are more 3-cliques than 2-cliques. The huge number of 3-cliques can be explained by the fact that most of the movies contained information about three actors, and these actors automatically formed a 3-clique. There were total 6 actors in the graph with degree 0, all of which were classified as 1-cliques.

Clique sizes and amounts in the actor graph

1: 6
2: 10
3: 5484
4: 68
5: 3

There could be found in total 7 different core sizes, varying from 2-core to 8-core. Core size was decreasing rapidly as the k increased: size of the 2-core was 6229 but size of the 8-core was not more than 84. Sizes of the different k-cores in the graph can be seen in the list below. The fact that there were bigger cores than cliques in the graph which can be explained by differences in these two concepts; a core is more loosely compounded than a clique.

Sizes of the different k-cores in the graph

2-core: 6229 nodes
3-core: 1785 nodes
4-core: 1316 nodes
5-core: 810 nodes
6-core: 522 nodes
7-core: 307 nodes
8-core: 84 nodes

7.5 Investigating how communities form

In task 5 we studied community forming by investigating cliques formed in our graph number 2, the movie graph. Our hypotheses was that cliques are formed on ground of genres (hypotheses 1) and that coverage of the dominating genre would be higher in smaller cliques (hypotheses 2), which means that the genre distribution in larger cliques would more resemble the original distribution.

There were in total 26 different genres mentioned in our data. Figure 3 below illustrates percentages of the dominating genre in each clique size compared to the total genre distribution. Green dots illustrate mean percentages of the dominating genre in different clique sizes. Note that the dominating genre can vary between cliques and the number of k-cliques is decreasing as k grows, so that the sampling is more reliable in case of smaller clique sizes. Red dots illustrate percentage of the dominating genre in the whole graph, which was drama (51.6 % of all movies). Some of the most general dominating genres in cliques were drama, action and comedy. As can be seen in the figure, the most common genre is more dominating in smaller cliques than in bigger ones, which supports our hypothesis number 2. Percentage of the dominating genre exceeds the total graph percentage also in bigger clique sizes, which supports out hypotheses number 1.
Based on these notations we can infer that cliques are to some extent formed based on genre. This is the case especially in the smaller cliques. 100% coverage of the dominating genre was nevertheless rare. It should be noted that genre descriptions in the dataset are done by several different persons and a movie belonging or not belonging to a specific genre is sometimes ambiguous. Genre borders in the dataset are also vague – for example “music” (4.32% of all movies) and “musical” (2.67% of all movies) are classified as two different genres.

Figure 3. Coverage of the dominating genre in cliques of different sizes. X-axis illustrates clique size and y-axis illustrates percentages of the dominating genre.

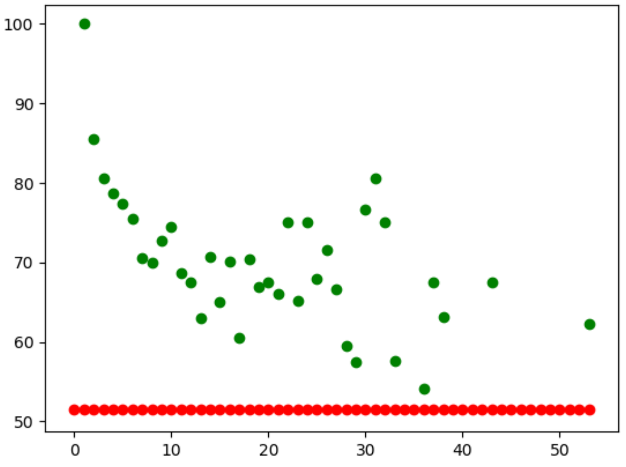
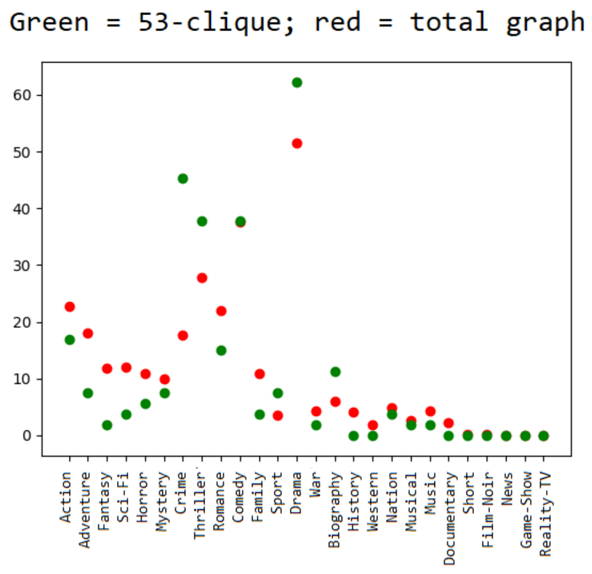


Figure 4. Example of two genre distributions: red dots illustrate percentages of genres in the total graph and green dots illustrate corresponding percentages in the biggest clique found in the graph, size 53.



7.6 Further community investigations

Language distribution of our movie graph was dominated by English (93.2%), followed by French (1.48%) and Spanish (0.81%). There were in total 47 languages present in our database. A remarkable portion of cliques – for example 923 of 962 3-cliques, 35 of 36 10-cliques and 7 of 8 15-cliques –

found in the graph contained only one language, mostly English. Some of the cliques included three different languages – for example English, Korean and Mandarin or English, Italian and French in 3-cliques. Based on these notations it seems clear that language is one remarkable factor in clique forming.

A common director seems not to be an explaining factor, which is understandable as there were over 2000 directors in the movie database containing roughly 5000 movies – in average one director for each 2,5 movies. It was rare that even two movies in a clique had been directed by a same person.

The third inspected factor was country in which the movie was filmed. There were in total 64 different countries mentioned in our database. The most common of them was USA (75.5% of all movies), followed by UK (8.84%) and France (3.14%). We inspected how many of the movie cliques was completely filmed in the same country. The percentages were surprisingly low. Only 599 of 962 3-cliques, 110 of 216 5-cliques and 15 of 36 10-cliques were completely filmed in the same country. The biggest clique, size 53, contained movies filmed in 5 different countries (USA, UK, Panama, Spain and Italy).

Based on these notations we can conclude that production country is not a remarkably explaining factor in clique formation, but language is. This seems surprising at the first glance, but further examination reveals that in fact many of the mentioned countries are English-speaking – cliques formed by movies filmed in such country combinations as USA, UK and Australia were not rare.

7.7 Actor Score

The actor scores were min-max normalized for further analysis. On average, actors had a score of *0.44*. In Figure 5, we have plotted the actor rankings against their score. From the flat middle section, we can see that for most part, the score between each rank is at an equal distance. However, at both ends, there are sharp, exponential slopes. This means that the difference between scores is more extreme at both ends. For example, the difference between ranks 100 and 200 is 6.6%, while the difference between ranks 2000-2100 is only 1.3%, and the difference between ranks 3000-3100 is 1.2%. The difference between ranks 6000-6100 behaves the same way as the top end, with a difference of 32%. In other words, there are a few actors who have great scores, a few actors who have a terrible score, and for the most part, there is a large middle section of actors with average scores.

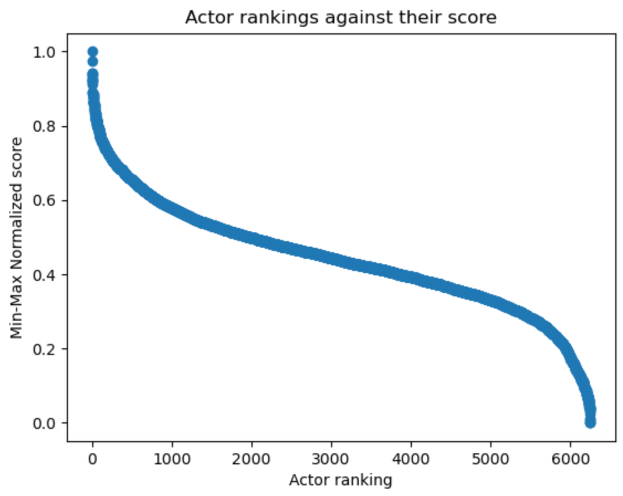


Figure 5. Actor ranking plotted against their score. Notice the sharp turns at the ends.

From Figure 6. we can see, that playing in more movies correlates with the score of the actor, but the ranking is not solely based on it. For example, some actors with who have played in 10 movies have higher ranking than actors who have played in 30. Scores seem to follow exponential growth, which might reflect the fact that part of the score is calculated by taking the logarithm of voter count.

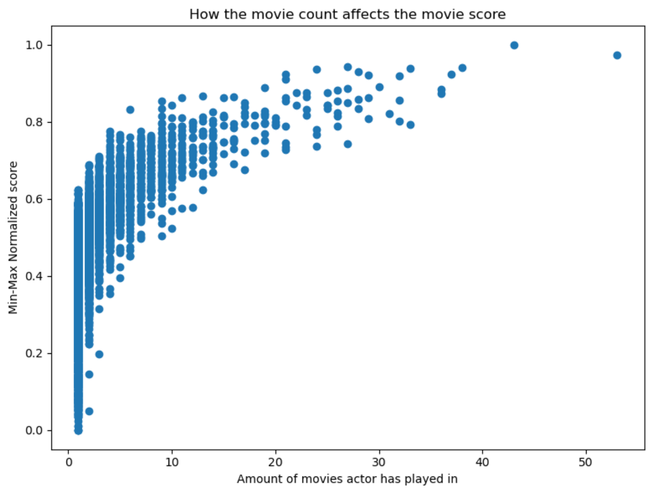


Figure 6 Plot of actor score against the number of movies the actor has been in. Playing in more movies seems to increase the score the actor can get, and to tighten the range of possible scores.

7.8 Actor ranking and centrality measures

The histograms for eigenvector centrality, betweenness centrality and actor ranking are shown in Figure 7. The histograms have been distributed into 100 bins. As can be seen from this, both centrality values are power law distributed, while actor rank follows bell curve distribution. Hence, neither of the centrality values are close to the actor rankings. The distances are shown in Table 2. It can be said that the eigenvector centrality is closer to the actor rankings, but that is probably because it is not as power law distributed as the betweenness centrality is, meaning that most of the actors have a very low centrality values, while few have high values.

Table 2. L1 distances for different centrality measures. L1 results are on range 0-2, where 0 is an exact match, and 2 means that the histograms do not overlap at all.

Centrality measure	L1 distance to actor values, range 0-2
Eigenvector centrality	1.882
Betweenness centrality	1.932

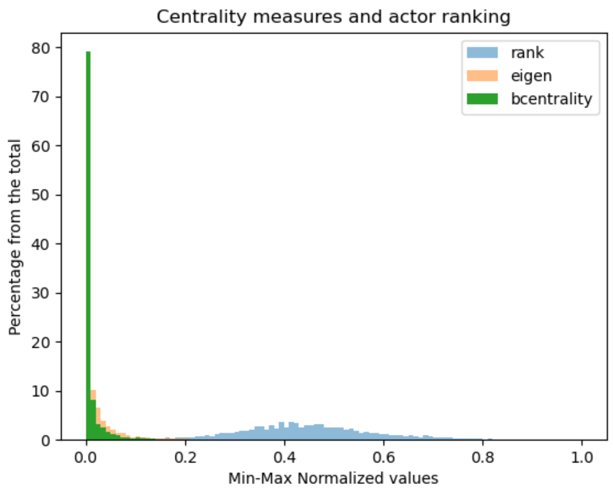


Figure 7. Histograms of actor rankings, actor eigen vector centrality and betweenness centrality values. All histograms have 100 bins and are min-max normalized.

The relationship between eigenvector centrality and actor score can be examined in more detail by plotting them against each other, like in Figure 8. While the L1-distance between the histograms is not great for these values, it does seem like high centrality values are only present in actors who have a high actor score. From this could be said, that if actor has a high eigen vector centrality value, they probably have a high actor score.

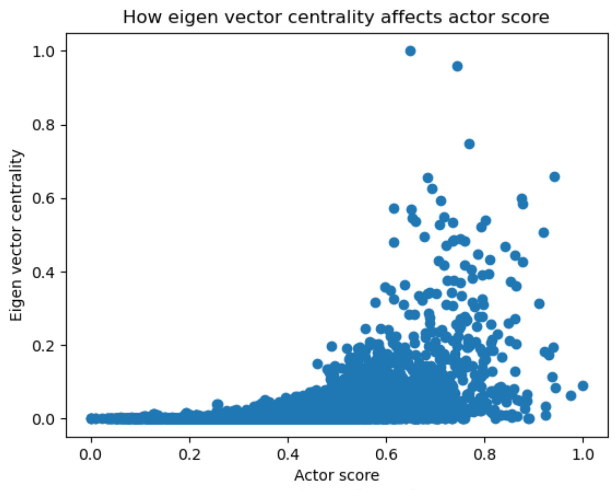


Figure 8. Eigen vector centrality plotted against actor score. The values on both axes are min-max normalized. Higher eigen vector centrality values are present only in actors who have a high actor score.

7.9 Random graph

In task 9 we created different random graphs in order to investigate how changes in the different parameters affect the histograms. In this section we present 5 of these random graphs.

For the first graph, we used function nx.gnp_random_graph(n, p) with parameters n = 6255 and p = 0.0007329. This p is calculated by dividing the amount of edges in the original graph by the highest possible edge amount in it. Comparison of some main properties of the random graph and original can be seen below in Table 3.

Table 3. Properties of random graph 1 compared with the original one.

	Original	Random graph

	graph	
Number of nodes	6255	6255
Average degree	4.58	4.61
Clustering coeff.	0.71	0.00056
Giant component	4627	0.00056
Diameter	15	12
Avg. path length	4.86	5.89

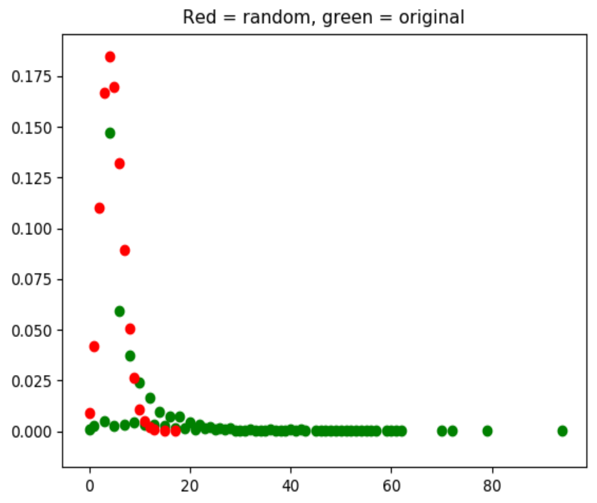


Figure 9. A graph illustrating degree distribution in the original graph and random graph number 1. X-axis represents degree of the node and y-axis represents the percentage of this degree.

As can be seen in Figure 9, average degree in the resulting random graph was close to the original but clustering coefficient was too small. The degrees of nodes in this random graph were normal distributed so that the random graph histogram was completely missing the highest degrees.

For the second random graph we used nx.powerlaw_cluster_graph(n, m p) to create a random graph.
We first utilized this function with parameters n = 6255, m = 2 and p = 0.2. Results can be seen in Table 4.

Table 4. Properties of random graph 2 compared with the original one.

	Original graph	Random graph
Number of nodes	6255	6255
Average degree	4.58	4.00
Clustering coeff.	0.71	0.16
Giant component	4627	6255
Diameter	15	9
Avg. path length	4.86	4.81

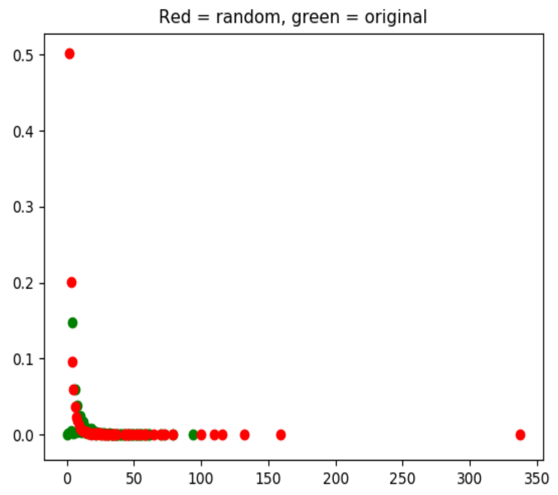


Figure 10. A graph illustrating degree distribution in the original graph and random graph number 2. X-axis represents degree of the node and y-axis percentage of this degree.

In the results it can be seen that the clustering coefficient was too low again, even though average shortest path and average degree were quite close to the original graph. The two degree distributions were quite similarly formed but there were a few outliers in the random graph distribution which the original distribution did not have.

For the third graph we utilized powerlaw_cluster_graph again with parameters n = 6255, m = 2 and p = 0.7. N and m remained the same, but p was increased from 0.2 to 0.7 in order to increase the clustering coefficient. Now the resulting random graph had a better clustering coefficient, even though still remarkably lower than in the original. The results are listed in Table 5.

Table 5. Properties of random graph 3 compared with the original one.

	Original graph	Random graph
Number of nodes	6255	6255
Average degree	4.58	4.00
Clustering coeff.	0.71	0.52
Giant component	4627	6255
Diameter	15	11
Avg. path length	4.86	5.04

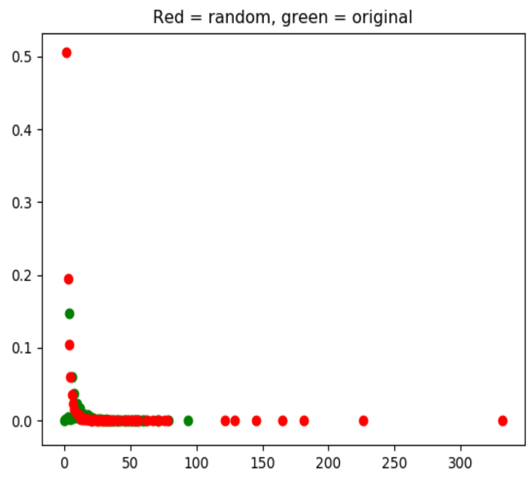


Figure 11. A graph illustrating degree distribution in the original graph and random graph number 3. X-axis represents degree of the node and y-axis percentage of this degree.

For the fourth graph we utilized powerlaw_cluster_graph again with parameters n = 6255, m = 2 and p = 0.95. Again, n and m remained the same, but p was again increased from 0.7 to 0.95 to further increase the clustering coefficient. Now the clustering coefficient of the random graph was satisfactory. This was reasonable because p stands for *"probability of adding a triangle after adding a random edge"* and in the original database most of the movies had 3 actors, which were connected with each other and this lead to a triangle being formed in most of the cases. The results are listed in Table 6.

Table 6. Properties of random graph 4 compared with the original one.

	Original graph	Random graph
Number of nodes	6255	6255
Average degree	4.58	4.00
Clustering coeff.	0.71	0.71
Giant component	4627	6255
Diameter	15	13
Avg. path length	4.86	5.60

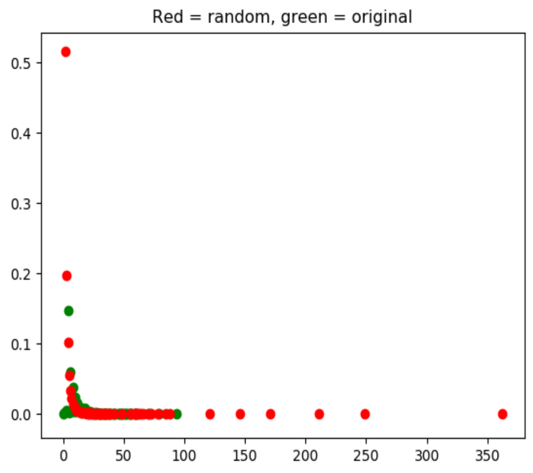


Figure 12. A graph illustrating degree distribution in the original graph and random graph number 4. X-axis represents degree of the node and y-axis percentage of this degree.

Now the two histograms started to look like each other, but still some disturbing outliers in the random graph were affecting the distribution too much. For the fifth graph we utilized powerlaw_cluster_graph again with parameters n = 6255, m =3 and p = 0.98. P was increased from 0.95 to 0.98 and m from 2 to 3 and furthermore we decided to remove 5 nodes with the highest degrees from the random graph. This turned out to alter the random graph disconnected – previous random graphs 2 – 4 were connected even though the original graph was not. Removing some of the nodes made the average clustering coefficient go back down again, but the distribution histograms were more similar. The results are listed in Table 7.

Table 7. Properties of random graph 5 compared with the original one.

	Original graph	Random graph
Number of nodes	6255	6255
Average degree	4.58	5.52
Clustering coeff.	0.71	0.53
Giant component	4627	6250
Diameter	15	12
Avg. path length	4.86	5.97

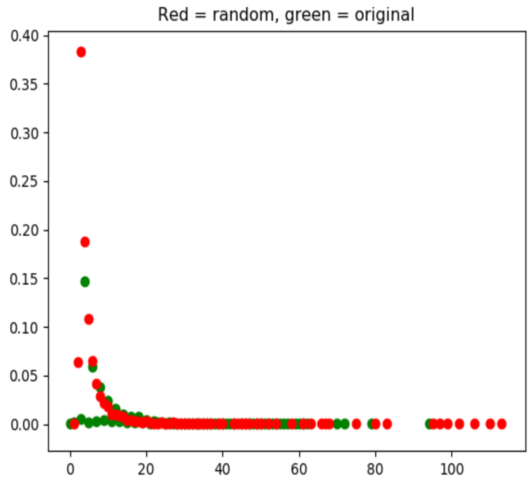


Figure 13. A graph illustrating degree distribution in the original graph and random graph number 5. X-axis represents degree of the node and y-axis percentage of this degree.

The original graph was clearly power-law distributed and could not be simulated with a normal distributed random graph. Using the random power law graph generator of networkx led to the random graph being fully connected and also containing some outliers with degrees much higher than the maximum nodes in the original graph. Removing some of the maximum degree nodes made the random distribution more resemble the original graph. Still the random generated distribution seems smoother than the original.
The random graph parameter p, ”probability of adding a triangle after adding a random edge”, needed to be raised very high (0.98) in order to increase the clustering coefficient high enough. This was due to the character of the original graph which includes remarkably many triangles.

A. 7.10 Literature

As the final task, we did a literature review of film making to find papers which support our results. We mostly searched papers from the Journal of Cultural Economics, which studies arts and culture from economical perspective.
No-one knows exactly how to create a hit movie. The success of a film relies on multiple factors such as the studio, which actors are in it, director, marketing and at what time the movie is launched. Albert S. argues that movie stars can be used as an indicator of the financial success of a film [18]. According to Albert, movie starts are one of the most stabile indicators of success for a movie, and that actors together with other unknown elements create a successful film. He describes a stochastic process where *1 / i (i + 1)* of film types produce a success, e.g. 50% of film types produce one successful film, 16.7% produce two etc. Film actors can be used as a stable indicator of the film type. This might help to explain the reason why only a fraction of the actors had such a large degree count, compared to the low average. However, others have argued that so called star power does not actually affect significantly the success of a movie [19]. As they say in Hollywood “nobody knowns anything” when it comes to making successful films.

VIII Conclusion and Perspectives

We created a graph of actors from IMDb 5000 movie database and analyzed the results in various ways. Each movie listed three actors, who were chosen randomly for each movie. The result was a rather disjoint graph, with multiple communities of three actors who did not join the giant component. We can only imagine what the graph would have looked if the movies listed more actors. The graph would probably been more connected. It would be interesting to repeat this research with a better dataset.
From these results, it seems that the top actors have played in most movies and are the most connected in this graph. They also seem to be older men, who

have been in the film industry for decades. This is no surprise, the older you are, the more movies you can make. However, most actors had on average two or three movies to their name in this list. This probably differs from their actual movie repertoire, but smaller titles were not on the list. The average budget of films on this list was 35 million, which is high enough to cut off small budget movies from the list.

References

[1] IMDb. URL: https://help.imdb.com/article/imdb/general-information/what-is-imdb/G836CY29Z4SGNMK5?ref_=helpsect_cons_1_1#. Retrieved 27.4.2020.
[2] Imdb statistics. URL: https://www.imdb.com/pressroom/stats/. Retrieved 27.4.2020.
[3] KAGGLE. URL: https://www.kaggle.com/tmdb/tmdb-movie-metadata. Retrieved 17.4.2020.
[4] ACTOR. URL: https://dictionary.cambridge.org/dictionary/english/lead?q=lead_10. Retrieved 17.4.2020.
[5] NetworkX. URL: https://networkx.github.io. Retrieved 17.4.2020.
[6] Algorithms – NetworkX 2.4 documentation. URL: https://networkx.github.io/documentation/stable/reference/algorithms/index.html. Retrieved 17.4.2020.
[7] Despalatovic, L., Vojkovic, T. & Vukicevic, D. 2014. Community structure in networks: Girvan-Newman algorithm improvement. MIPRO 2014, 26-30 May 2014, Opatija, Croatia. URL: http://www.ccs.neu.edu/home/vip/teach/DMcourse/6_graph_analysis/notes_slides/06859714.pdf. Retrieved 27.4.2020.
[8] Community API – Community detection for NetworkX 2 documentation. URL: https://python-louvain.readthedocs.io/en/latest/api.html. Retrieved 17.4.2020.
[9] Communities – NetworkX 2.4 documentation. URL: https://networkx.github.io/documentation/stable/reference/algorithms/community.html. Retrieved 27.4.2020.
[10] networkx.algorithms.clique.find_cliques – NetworkX 2.4 documentation. URL: https://networkx.github.io/documentation/stable/reference/algorithms/generated/networkx.algorithms.clique.find_cliques.html. Retrieved 17.4.2020.
[11] networkx.algorithms.core.k_core – NetworkX 2.4 documentation. URL: https://networkx.github.io/documentation/stable/reference/algorithms/generated/networkx.algorithms.core.k_core.html. Retrieved 17.4.2020.
[12] Sung-Hyuk Cha, Sargur N. Srihari. On measuring the distance between histograms. Pattern Recognition 35, 1355-1370 (2002)
[13] gnp_random_graph – NetworkX 1.10 documentation. URL: https://networkx.github.io/documentation/networkx-1.10/reference/generated/networkx.generators.random_graphs.gnp_random_graph.html. Retrieved 27.4.2020.
[14] powerlaw_cluster_graph – NetworkX 1.10 documentation. URL: https://networkx.github.io/documentation/networkx-1.10/reference/generated/networkx.generators.random_graphs.powerlaw_cluster_graph.html. Retrieved 27.4.2020.
[15] Brandes, U., Delling, D. et al. 2008. On Modularity Clustering. IEEE Transactions on Knowledge and Data Engineering. Volume: 20, Issue 2, Feb. 2008. URL: https://www.uni-konstanz.de/mmsp/pubsys/publishedFiles/BrDeGa08.pdf. Retrieved 27.4.2020.
[16] networkx.algorithms.community.quality.coverage - NetworkX 2.4 documentation. URL: https://networkx.github.io/documentation/stable/reference/algorithms/generated/networkx.algorithms.community.quality.coverage.html. Retrieved 27.4.2020.
[17] networkx.algorithms.community.quality.performance - NetworkX 2.4 documentation. URL: https://networkx.github.io/documentation/stable/reference/algorithms/generated/networkx.algorithms.community.quality.performance.html. Retrieved 27.4.2020
[18] Albert, S. Movie Stars and the Distribution of Financially Successful Films in the Motion Picture Industry. Journal of Cultural Economics 22, 249–270 (1998).
[19] De Vany, A., Walls, W.D. Uncertainty in the Movie Industry: Does Star Power Reduce the Terror of the Box Office?. Journal of Cultural Economics 23, 285–318 (1999).

APPENDIX

APPENDIX 1. The 10 most central actors according to degree centrality.

Actor, degree centrality

1. Robert De Niro, 94
2. Morgan Freeman, 79
3. Matt Damon, 72
4. Bruce Willis, 70
5. Steve Buscemi, 62
6. Nicolas Cage, 62
7. Johnny Depp, 61
8. Liam Neeson, 60
9. Will Ferrell, 60
10. Denzel Washington, 60

APPENDIX 2. The 10 most central according to eigenvector centrality.

Actor, eigenvector centrality

1. Robert De Niro, 0.2014000381185906
2. Morgan Freeman, 0.19349174428033303
3. Brad Pitt, 0.15059648651051707
4. Johnny Depp, 0.13258456445409808
5. Matt Damon, 0.13182915750614493
6. Bruce Willis, 0.12626057281800754
7. Kirsten Dunst, 0.1205557379871974
8. Anthony Hopkins, 0.11965478172872412
9. J.K. Simmons, 0.11766520165864047
10. Meryl Streep, 0.11537218709036129

APPENDIX 3. The 10 most central actors according to betweenness centrality.

Actor, betweenness centrality

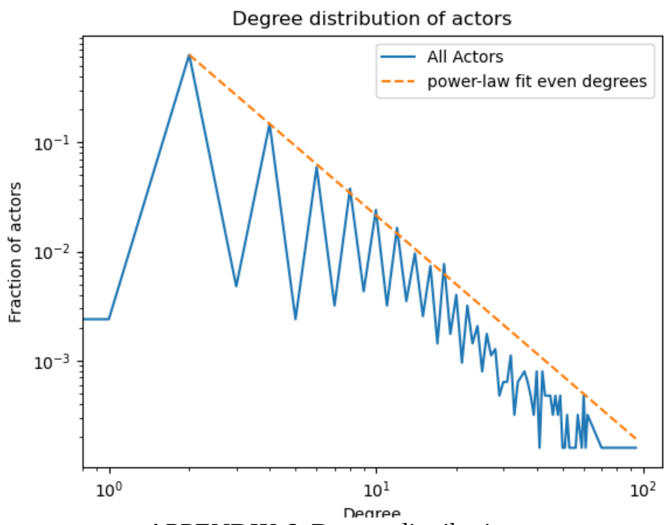
1. Robert De Niro, 0.030640372631812008
2. Morgan Freeman, 0.02376092558045719
3. Bruce Willis, 0.01897514966346706
4. Matt Damon, 0.017833539397414066
5. Denzel Washington, 0.017265317561134082
6. Nicolas Cage, 0.016077657112834483
7. Bill Murray, 0.014633007043050619
8. Will Ferrell, 0.014177727878688349
9. Steve Buscemi, 0.013240476035766316
10. Jim Broadbent, 0.012929955960287976

APPENDIX 4. The 10 most central actors according to closeness centrality.

Actor, closeness centrality

1. Robert De Niro, 0.23461105395074952
2. Morgan Freeman, 0.234241638311613
3. Matt Damon, 0.22557803341361624
4. Bruce Willis, 0.2250735173541462
5. Johnny Depp, 0.2243503777337703
6. Brad Pitt, 0.22424745122674863
7. Denzel Washington, 0.22398321620959222
8. Kirsten Dunst, 0.22270031312273697
9. Steve Buscemi, 0.2225409828174134
10. Meryl Streep, 0.22173336859683307

APPENDIX 5. Distribution of Degrees on logarithmic scale. Note that the power-law fitting starts from 2, and not from 0 even though it is an even number. From this figure it can be seen that the low sample count shows up as noise on the tail end.



APPENDIX 6. Degree distributions

degree 0 count 6 fraction 0.00096
degree 1 count 15 fraction 0.0024
degree 2 count 3937 fraction 0.62992
degree 3 count 30 fraction 0.0048
degree 4 count 921 fraction 0.14736
degree 5 count 15 fraction 0.0024
degree 6 count 369 fraction 0.05904

degree 7 count 20 fraction 0.0032
degree 8 count 235 fraction 0.0376
degree 9 count 27 fraction 0.00432
degree 10 count 150 fraction 0.024
degree 11 count 20 fraction 0.0032
degree 12 count 103 fraction 0.01648
degree 13 count 22 fraction 0.00352
degree 14 count 60 fraction 0.0096
degree 15 count 16 fraction 0.00256
degree 16 count 46 fraction 0.00736
degree 17 count 9 fraction 0.00144
degree 18 count 48 fraction 0.00768
degree 19 count 11 fraction 0.00176
degree 20 count 25 fraction 0.004
degree 21 count 6 fraction 0.00096
degree 22 count 20 fraction 0.0032
degree 23 count 9 fraction 0.00144
degree 24 count 13 fraction 0.00208
degree 25 count 5 fraction 0.0008
degree 26 count 11 fraction 0.00176
degree 27 count 7 fraction 0.00112
degree 28 count 8 fraction 0.00128
degree 29 count 3 fraction 0.00048
degree 30 count 4 fraction 0.00064
degree 31 count 4 fraction 0.00064
degree 32 count 7 fraction 0.00112
degree 33 count 2 fraction 0.00032
degree 34 count 4 fraction 0.00064
degree 36 count 5 fraction 0.0008
degree 37 count 4 fraction 0.00064
degree 38 count 3 fraction 0.00048
degree 39 count 2 fraction 0.00032
degree 40 count 5 fraction 0.0008
degree 41 count 1 fraction 0.00016
degree 42 count 5 fraction 0.0008
degree 43 count 3 fraction 0.00048
degree 45 count 3 fraction 0.00048
degree 46 count 2 fraction 0.00032
degree 47 count 3 fraction 0.00048
degree 48 count 2 fraction 0.00032
degree 49 count 3 fraction 0.00048
degree 50 count 1 fraction 0.00016
degree 51 count 1 fraction 0.00016
degree 52 count 2 fraction 0.00032
degree 53 count 1 fraction 0.00016
degree 54 count 1 fraction 0.00016
degree 55 count 1 fraction 0.00016
degree 56 count 1 fraction 0.00016
degree 57 count 2 fraction 0.00032
degree 59 count 1 fraction 0.00016
degree 60 count 3 fraction 0.00048
degree 61 count 1 fraction 0.00016
degree 62 count 2 fraction 0.00032
degree 70 count 1 fraction 0.00016
degree 72 count 1 fraction 0.00016
degree 79 count 1 fraction 0.00016
degree 94 count 1 fraction 0.00016

APPENDIX 7. Top 10 and bottom 10 actors based on actor scoring, min-max normalized.

1: Morgan Freeman 1.0000
2: Robert De Niro 0.9738
3: Christian Bale 0.9431
4: Bruce Willis 0.9419
5: Brad Pitt 0.9385
6: Robert Duvall 0.9359
7: Tom Hanks 0.9298
8: Al Pacino 0.9238
9: Matt Damon 0.9228
10: Harrison Ford 0.9220
…
6241: Jemima West 0.0390
6242: Gary Daniels 0.0347
6243: Matt Marr 0.0347
6244: Ginger Jensen 0.0347
6245: Maureen McCormick 0.0241
6246: Mike Beckingham 0.0105
6247: Tom Stedham 0.0105
6248: Scott Levy 0.0000
6249: Vanilla Ice 0.0000
6250: Jennifer Sky 0.0000