Movie Casting Problems

Alireza Torabian
torabian@yorku.ca
York University
Toronto, Ontartio, Canada

Hossein Naderibeni
hn268@yorku.ca
York Universiry
Toronto, Ontartio, Canada

KEYWORDS

actor suggestion, co-star suggestion, link weight prediction, graph autoencoder, movie, genre, actor, graph, network

1 ABSTRACT

Movie producers and directors need to audit with many actors to find and choose the best fits for their movies, a time-consuming and challenging process. Since there are many rich movie-related datasets available on the web, such as the movies dataset from Kaggle, we can use them to help movie producers in this process. The main tasks we have tried to solve are suggesting the best alternative actors and best co-actors, rating the relationship between a director and an actor, and comparing two casts to play in a movie. We have achieved high-quality results in the mentioned problems except director-actor rating. The main weakness of our director-actor approach was modeling based on a wrong assumption, which is described in the evaluation section. We have used a graph autoencoder called Local Neighborhood Graph Autoencoder (LoNGAE) on a network named Actors Network created using the ratings in the used dataset. Our model has achieved the mean-squared error as low as 0.005406 on link weight prediction in Actors Network with edge weights between 0 and 1. The main challenge in this project was the evaluation part. Since there is not much available data for the mentioned tasks, we have used related information available on movie news or posts on the web to have an intuitive evaluation. Reference code is available at https://github.com/1997alireza/Movie-Casting-Problems.

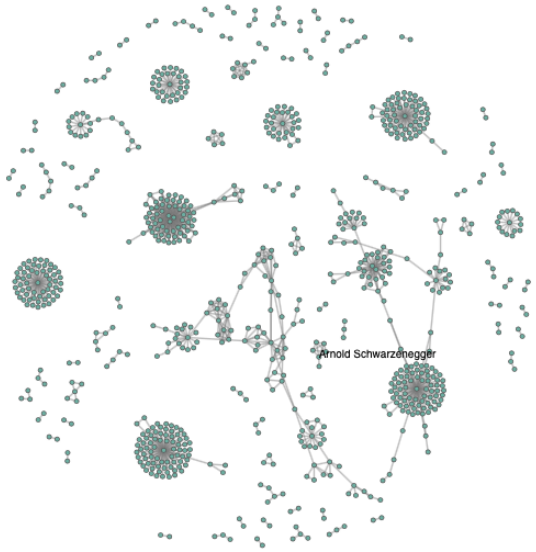

Figure 1: Actors Common Movie Network: Each actor has shown with a node and two actors are adjacent if they have played in a movie together.

2 INTRODUCTION

There are many works done on movie datasets based on users' needs, such as movie recommendations. However, one of the cinema industry's significant needs is to help producers choose appropriate casts for the product he wants to produce. Using the extensive data available about movies, producers can have more reasonable choices; however, it would be hard for them to follow this data which is increasing every day. Our focus is on producers' problems related to movie casting. It is important to know which actors are better for a movie and which actors can work better together. Casts of a movie play an essential role in the success or failure of a movie. Currently, they have to know lots of different actors suitable for each role to audition with. This matter made us think about these problems and try to answer them using state-of-the-art approaches on movie datasets. When a producer decides to cast an actor/actress for a movie, there are many difficulties. He should find candidate actors for each movie's roles, which depend on many factors such as the movie genre. He also needs to assess each director-actor and actor-actor relations. These relations can be checked out through their previous works together or with other casts. One another common problem between directors is to find an alternative actor when their choice is not available. The complicated problem is that whether a cast group works out for a movie. We tried to solve some of these problems using graph based and neural network based approaches.

The neural network based model used in this project is a graph autoencoder called Local Neighborhood Graph Autoencoder (LoNGAE) [11], whose objective is to jointly learn the latent representation that can reconstruct both the input graph's weights and the node features. We apply this model on the network of relationships between actors.

The only dataset we have used in this project is the movies dataset from Kaggle [2]. We have done some analyses on the movies dataset during the project, and some of the characteristics of a network between actors extracted from this dataset. As shown in figure 1, there are many complete components, and few actors have a high degree.

As mentioned earlier, movie-related data is too big to handle with human eyes, and we are going to find intuitions to help producers and directors. They will decide on their final choices, but giving

them data-based hints may help them. In the next section, the main problems that we have focused on are explained.

3 PROBLEM DEFINITION

3.1 Alternative Actor Suggestion

Sometimes when a producer or director chooses an actor for a role in their movie, the actor is not available or does not accept the role. In such a situation, they would be looking for an alternative actor. The problem here is finding which actors can be acceptable alternatives. We try to calculate a score between actors showing how much similar they are and can replace each other. Using these scores, a list of alternative actors is provided for producers.

3.2 Director-Actor Rating

A producer would need a lot of effort to find his/her casts and crews to be the best to make a movie. One of the challenges is to know how much a director matches with an actor. After a producer chooses his movie director, they need to gather a group of actors together. But how can they find the best matches among maybe millions of available actors? Usually, they would choose actors who know before or have had movies together; however, it would not be a suitable approach for a great producer and director. According to the massive available data from datasets such as the movies datasets [2], it should be easier to solve this problem for producers. How can we predict the best matches between directors and actors using available movie data?

3.3 Co-Star Suggestion

Movies have about for main actors in general. The relation between the main actors plays a key role in the success of the movie. Therefore, in addition to the importance of actors' competence for the roles, the producer needs to know about the relation between actors. To help the producers with this problem, we generate a co-star suggestion list for each actor based on a score between them, showing how much fit they are in the movie's main roles together. On calculating this score, it is expected to consider the movie's attributes the producer chooses the actors for.

3.4 Movie Casts Rating

Suppose that a producer doubts which of two actors group is better for his movie. However, he knows each group's actors are perfect matches for each other; he cannot decide on one of them. We need an approach to measure the quality of a group of actors for a movie, considering its attributes. One of the applications of this task in this project is evaluating alternative actor suggestion task performance. Using the movie-cast-rating, we rate a group of actors with the original actor and the alternative actor. We did not use this rating to evaluate the co-star suggestion task since these two tasks use the same model in essence.

4 RELATED WORK

We could not find any specific work on how similar two actors are or how good they match each other to be co-actors based on their acting. Although there are many works done on how visually similar. In order to find how similar two movies are there are lots of works done on recommender systems which suggest users same movies to the movies they like, but our work is not user based. So we used a technique on how two movies are similar according to their content.

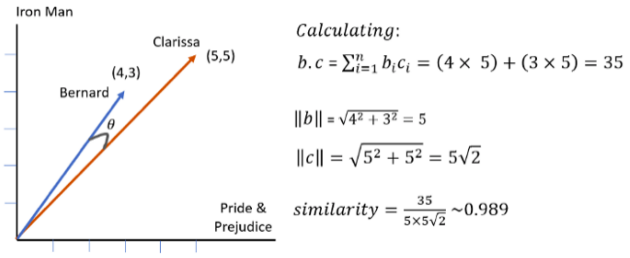

Figure 2: Cosine Similarity for two movies with vectors (4,3) and (5,5)

One of the common ways to find similar movies is to use cosine similarity[4]. We assign vectors to the movie features and try to calculate how close two vectors are and cosine similarity is a good way for this purpose.

The network between directors and actors can be seen as a bipartite graph. A bipartite graph is graph with two disjoint sets of nodes. These sets do not have any intraconnections, and each edge is between a node from the first set and a node from the second set. The edges of the director-actor graph shows the movies in which a director and an actor have worked together. Link prediction in bipartite graphs is different from regular graphs. General link predictions are mostly using the paths of length two between nodes. Since in a bipartite graph all paths between two adjacent vertices are of odd lengths, these general link predictions do not apply on our graph. Few new approaches have been proposed for link prediction in this type of graphs. Some of them uses random walks in the graph to estimate a score for a path between two heterogeneous nodes (two nodes that are not in the same set) [6]. Some algorithms use projection on the bipartite graph to transform it into a unipartite graph (See figure 3); In the projected graph, only the nodes of one of the node sets are used, and they are connected if they have common connected nodes in the original graph [1]. One popular link prediction approach, which can be used in both unipartite and bipartite graphs, is kernel-based approach [5].

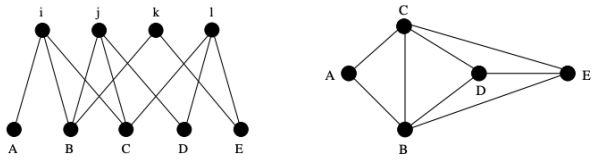

Figure 3: An example of bipartite graph G (left), and its ⊥-projection G⊥(right) [1].

5 METHODOLOGY

5.1 Actors Network

First of all, we create a graph named Actors Network showing the relation between movies' main actors. We only consider the first five actors of each movie because these are the actors who play the main roles of movies. Each node of the graph is related to one of these actors. Actors Network is a weighted undirected graph with edge weights calculated as follows (*c_movies* is the list of the common movies between a and b):

$$weight(a,b) = \frac{Average(rating(c\_movies))}{10}$$

The weight of an edge between two actors shows how successful their common movies have been. These weights are a measure to see how good two actors are to play in a movie together. Since movie ratings in the used dataset are in the range from 0 to 10, we divide the average rating by 10. Each node in this graph has a feature vector as well ($movie_{g_i}$ is the list of movies of a in the i-th genre):

$$feature(a) = Normalize \begin{bmatrix} Average(rating(movie_{g_1})) \\ ... \\ Average_i(rating(movie_{g_n})) \end{bmatrix}$$

Our models are able to recognize the attributes of related movies to each actor using node features to some extent.

5.2 Local Neighborhood Graph Autoencoder

Our idea to answer the mentioned problems in this project is to provide two primary models and use them for the mentioned tasks. The primary models we need are as follows:

- A link weight prediction model to do the prediction on Actors Network.
- A transformation from actors to a space such that similar actors are detectable in that space.

Using a graph autoencoder (GAE), we can solve both of these primary problems. An autoencoder is a neural network that tries to encode the input data into a latent space and use this space to reconstruct the input data. The latent space of a graph autoencoder is the space that we can measure the similarity between actors, and the output weights of this model can be used as the predicted link weights. Since the nodes in Actors Network have features, we need a graph autoencoder that captures both local features of each node, which are the weights of its edges, and the provided node features in our network. Therefore, we use local Neighborhood Graph Autoencoder (LoNGAE), whose objective is to jointly learn the latent representation that can reconstruct both the input graph and node features [11]. Each of the encoder and decoder contains two layers, and the latent space is 128-dimensional.

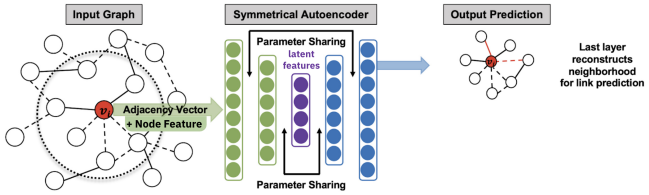

Figure 4: Local Neighborhood Graph Autoencoder

5.2.1 Link Weight Prediction. We have used the mean squared error as the loss function of the graph autoencoder for the graph weights. This part aims to reconstruct the original edges of the graph and predict weights for the missing edges that are not presented in there. Consequently, the mean squared error is only calculated on the edges presented in the input graph, and the model is free to predict weight for the missing edges. Our model has achieved the validation error 0.005406 on this task after 50 epochs.

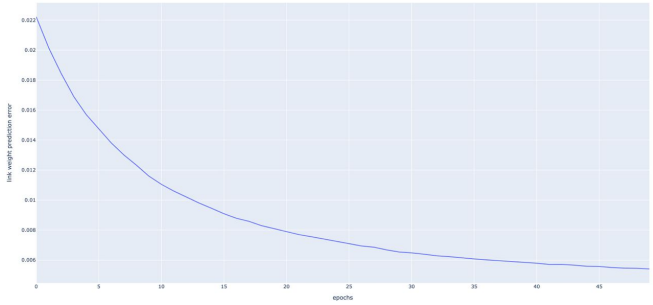

Figure 5: Link weight prediction validation error across 50 training epochs

Here we can see the distributions of the original and the predicted weights of the graph. They are almost in the same distribution.

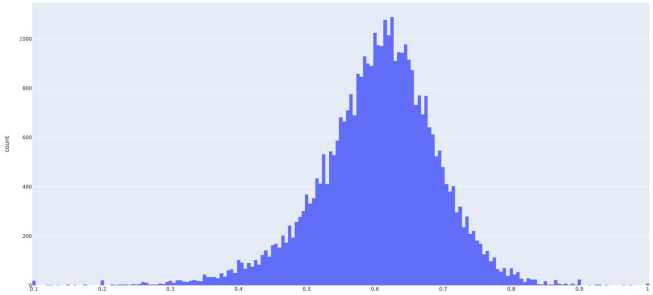

Figure 6: Original weights of the input graph

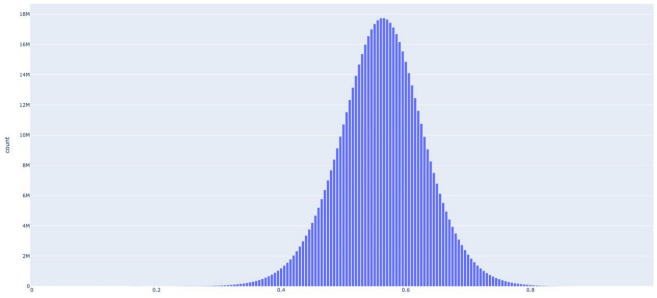

Figure 7: Predicted weights for the input graph

5.2.2 Node Feature Reconstruction. The loss function to reconstruct the node features using the graph autoencoder is weighted cosine similarity. Since the features are normalized vectors between zero and one, cosine similarity is an appropriate loss function. We see an uneven distribution of movies in genres in figure 8. Therefore, we use a weighted loss function to balance the effect of each genre.

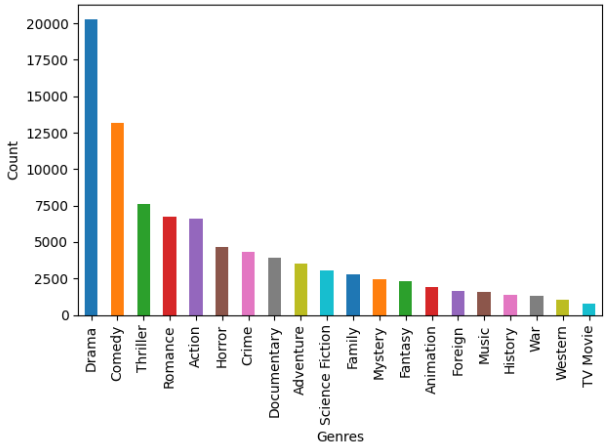

Figure 8: Distribution of genres in the movies dataset

5.3 Alternative Actor Suggestion: Movie Similarity Based

An actor can replace another actor in a movie if their type of acting is similar together. To find similar actors for auditioning, we tried to make a similarity rank for movies and then try to find their common similar movies. To find movies that look like each other with cosine similarity. We made a ranking for how two movies are close to each other based on features like genre & keywords. And if two actors have lots of similar movies in common, they are probably good alternatives for each other.

	0	1	2	3	4	5
0	1.00000	0.84000	0.83205	0.84000	0.80498	0.78087
1	0.84000	1.00000	0.72111	0.76000	0.62610	0.78087
2	0.83205	0.72111	1.00000	0.94299	0.86824	0.73635
3	0.84000	0.76000	0.94299	1.00000	0.80498	0.84334
4	0.80498	0.62610	0.86824	0.80498	1.00000	0.62859
5	0.78087	0.78087	0.73635	0.84334	0.62859	1.00000

Figure 9: Cosine Similarity Matrix for Movies: Toy Story, Jumanji, Grumpier Old Man, Waiting to Exhale, Father of the Bride Part II, Heat

Cosine similarity measures the similarity between two vectors using their inner product space by the cosine of the angle between two vectors and determines whether two vectors are pointing in roughly the same direction. Cosine Similarity distance is often used to measure document similarity in text analysis. For input of similarity rank, we used these movie's features: genres, keywords, language, adult rating from The Movies Dataset[2]. We combined these features of each movie and calculated cosine similarity between m. With the results, we made a bipartite graph of actors to movies that each actor has edges to the film he has played roles. We can compute the common movies of actors using this graph and using the similarity rank of movies we proposed earlier; we can calculate how much two actors are close to each other and played roles in similar movies.

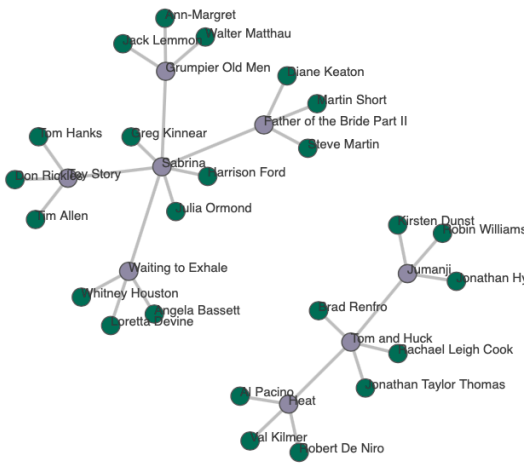

Figure 10: Actor Movie Network: Similar movies(grey nodes) are connected with edges and Actors(green nodes) who act in similar movies are considered as candidate alternatives.

5.4 Alternative Actor Suggestion: GAE Based

Using our trained GAE model, we have a high dimensional space named latent space, representing each input node's encoded features. Each actor is assigned to a feature vector in the latent space, and similar actors have similar features. A small portion of the latent space of the presented actors in our model has been shown

in the Figure 11. Since the latent space is 128-dimensional, we have used Principal component analysis (PCA) to project each vector into a 2-dimensional space; therefore, we could show this space in a plot. We can see some well-known close actors such as Henry Fonda and Morgan Freeman and Matt Damon and Brad Pitt in a small distance.

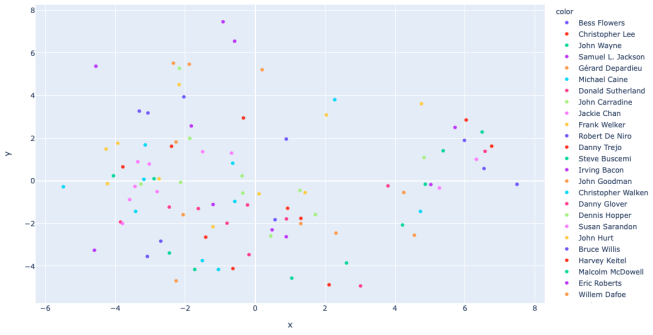

Figure 11: PCA reduction of the 100 most active actors' latent feature

We can use this to suggest similar actors based on the similarity of other vectors to it. On this point, two questions arise; How to measure distance on the vector space and how to find the closest vectors in a reasonable running time. Since there are around 10^5 actors and many queries, this may take some time using brute force search.

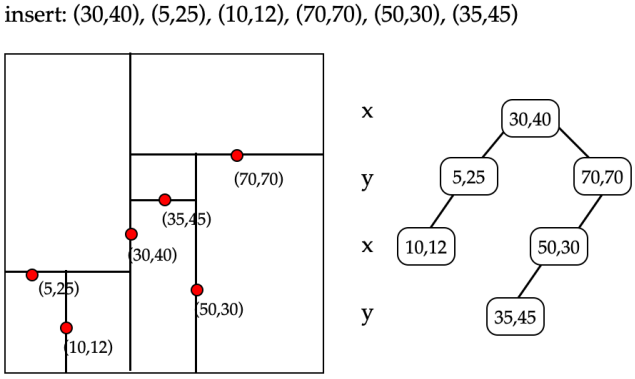

Figure 12: k-d tree example [8]

There are options as Manhattan, Euclidean, Minkowski, Chebyshev, or Cosine distance (discussed in 3.1) for the metric distance. We are going to measure euclidean distance here. Our problem in other languages is finding the nearest neighbors of a given point in a k-dimensional space. k-d tree (short for k-dimensional tree) is a space-partitioning data structure for organizing points in a k-dimensional space. In amortized analysis, insertion and search queries are $O(log\ n)$. Moreover, since we dump our k-d tree after one-time construction, this overhead will be gone. Here we also use `sklearn.neighbors.KDTree` implementation of k-d Tree form sci-kit in python.

5.5 Director-Actor Rating

The goal in this part is to predict that which directors and actors can work better together. Based on the available dataset that we have, we can rely on the ratings of the movies as a criterion of how much a director and an actor are appropriate for each other. Using this information we can say a director-actor pair is good if they have had movies with high ratings together. So it cannot solve our problem very well. We need to predict a good pair even they have not worked together yet. The relation between the directors and actors can be considered as a bipartite graph. This bipartite graph for a small portion of the dataset has been shown in figure 13.

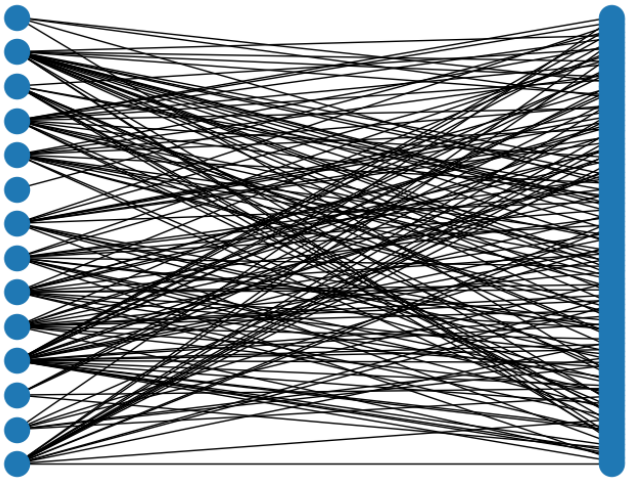

Figure 13: Bipartite graph between directors and actors. Left nodes are the directors, and right nodes are the actors connected to those directors.

Each edge in this graph shows that the connected director and actor have made one or more movies together. A weight has been added to each edge that is equals to the average of movies' ratings that those director and actor have had. The rating of each movie is calculated using the user's ratings in the dataset. The weight shows how well a movie could be if both director and actor are used. As it has been mentioned before, our task is to predict which directors and actors can make better movies together, which lead us to a link prediction task in the director-actor weighted bipartite graph. New suggested actors for a director are predicted using successful actors of similar directors. This task is done following these four steps:

(1) A feature for each director is calculated based on the graph. This feature contains the average movies' ratings of the director and all actors. If a director and an actor do not have any common movie, the related value in the feature map is zero. Actually, this feature is the director's row in adjacency matrix of the graph.

(2) The similarities between directors are calculated using their feature maps, which is calculated in step one. We use the cosine similarity to calculate the similarity $s_{i,j}$ between director i and j.
(3) For each director, we compute the cast list suggestion from the other directors as follows. $feature_i$ is the feature map of each director calculated at step one. We use the difference between the cast's ratings of directors as their suggestions. Then we eliminate negative values. A negative value shows that the director i has better movies with the related cast rather than the director j, so in this case director j cannot suggest or decline the cast for director i.

$$suggestion_{j-i} = max(feature_j - feature_i, 0)$$

(4) To achieve the final suggestion for director i, we sum the suggestions from all other directors. In this summation, the similarity between directors should be considered, so it is calculated as follows:

$$suggestion_i = \frac{\sum_{j \in directors}(suggestion_{j-i} * s_{i,j})}{len(directors)}$$

The higher values in the final vector for director i are suggesting more appropriate actors to achieve more successful movies.

5.6 Co-Star Suggestion

We use the predicted ratings by our graph autoencoder as the ratings show how good two actors are to play together. Using the given features of an actor, the predicted weights with all other actors in the graph are taken. Actors with higher weight are suggested as the best co-stars for the given actor.

5.7 Movie Cast Rating

To rate a cast for a movie, we need a criterion to measure the quality of the relationships between the given actors and another to see how much fit these actors are for the given movie. Our graph autoencoder provides both criteria for us by the predicted weights between actors and predicted node features. Since node features show the success of actors in each movie genre, we have the second required criterion by comparing node features with the genres of the given movie. We calculate a similarity score s and a weight w for each pair in the cast as follows:

$$s(a,b) = \frac{\text{predicted}(a\text{->}b) + \text{predicted}(b\text{->}a)}{2}$$

$$w(a,b) = \sum_{g \in \text{genres}_m} feature_g(a) + feature_g(b)$$

In these equations, a and b are actors, $genres_m$ is the genres' list of the given movie. Since our model cannot recognize the difference between directed and undirected graphs, it generates two separate weights for each edge in each direction. So we use the average of the generated weights for an edge in two directions as the similarity factor between two actors.

In the end, the rating of the given cast is calculated using the equation below. It is the weighted average of pairs similarity scores with the weights w:

$$\text{cast\_rating} = \frac{\sum_{a,b \in cast} s(a,b) * w(a,b)}{\sum_{a,b \in cast} w(a,b)}$$

6 RESULTS

6.1 Alternative Actor Suggestion

In this section, we will see the results of the two alternative actor methods we said earlier and compare them. First of all, one major problem evaluating is that we do not have any test data. There is not any scalar dataset indicating how much two actors are a good pair as replacements. So we are going to solve this problem by two approaches. First, we will change an actor for a given movie according to alternative actor methods suggestions and then compute the difference of movie-cast rating. If we assume movie-cast rating is a reliable parameter, it can be used to compare the new cast with the old one. The second approach is to use data from movies. We have run methods on some actors(see Table 1, more complete version is included in lecture slides)

The results of the first test show us that the second method works better. However, somehow, some of the first method results are better. There is a significant difference here, and it is that the second approach is like a black box and does not tell us about the decision process. Nevertheless, the first method tells us similar movies between two actors, and we can fact check whether this is a good pair. Another important property that we have to consider is the fast running time of the second method. However, each method has its pros and cons, but the second one is more practical because of running time.

Actor	Movie Similarity suggestion	GAE suggestion
Mel Brooks	Steve Martin	Jack Lemmon
Al Pacino	Sylvester Stallone	Trevor Howard
Tom Hanks	Bridget Fonda	Tony Curtis

Table 1: Alternative Actor Methods Comparison

To compare two actors, we have used a radar chart showing actors' ratings in each genres. As we can see alternative actors are similar to the original actors in these charts:

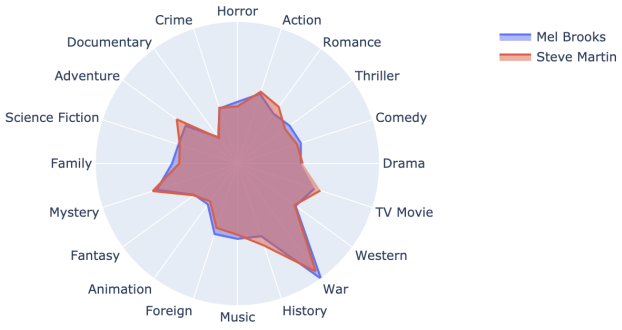

Figure 14: Radar chart of Mel Brooks and his suggested alternative

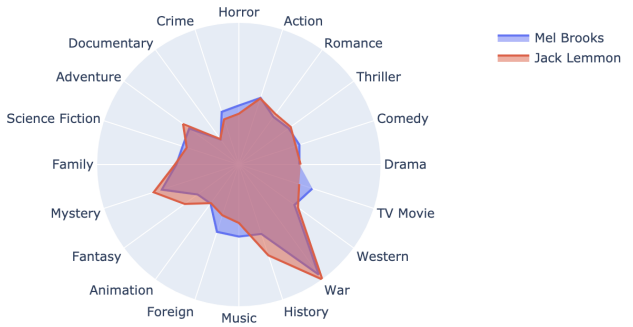

Figure 15: Radar chart of Mel Brooks and his suggested alternative

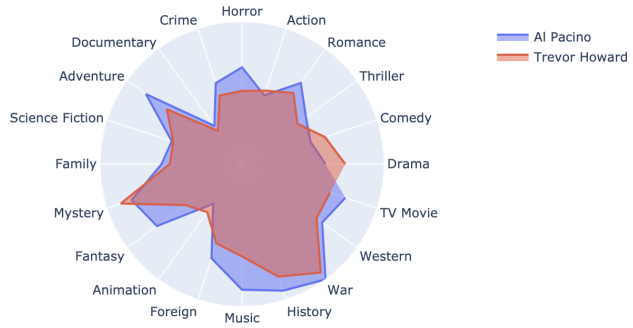

Figure 16: Radar chart of Al Pacino and his suggested alternative

6.2 Director-Actor Rating

To evaluate the link prediction model, first, we use 25% of the original dataset to create a director-actor graph. Then we use this graph for link prediction and extract top suggestions with higher scores between all directors and actors. Then we evaluate based on the actual high rating edges obtained from the other portion of the original dataset, which is 75% of the dataset. We have selected the top 0.1% predictions and the edges in the second graph with the average rating equals or more than 4. The portion of actual high rating edges that have been predicted was close to zero, which shows that the prediction could not achieve our goal to solve the problem. This evaluation shows that we cannot determine the quality of the relationship between a director and an actor based on similar directors. Two directors could have the same ratings with some actors but different relation and rating with another actor. Another issue in this prediction that can take into account is the approach of weighting the edges between each director and actor. A high rating of a movie depends on many factors, and we cannot rate the relation between two people in a group because the group has achieved a good result.

6.3 Co-Star Suggestion

To analyze the results of the Co-Star Suggestion, we used lists of some famous actor duos in cinema history from the websites [3] and [7] as our test actor pairs. The predicted weights between these pairs from our model are provided in table 2. The distribution of all predicted weights between actors is shown in figure 17. The red portion in this figure is the range that most of the predicted weights between our test actor pairs belong to.

Duo	Predicted Weighting
Chris Evans, Scarlett Johansson	0.6687
Anne Hathaway, Jake Gyllenhaal	0.7095
Ryan Gosling, Emma Stone	0.6432
Steve Carell, Christian Bale	0.6629
Will Ferrell, John C. Reilly	0.5880
Bradley Cooper, Jennifer Lawrence	0.6485
Johnny Depp, Helena Bonham Carter	0.6566
Al Pacino, Robert De Niro	0.6995

Table 2: Co-Star ratings for some famous actor duos

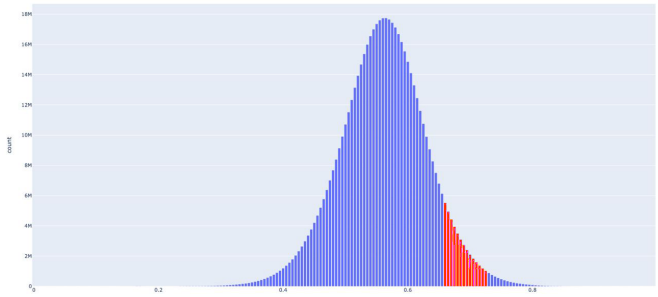

Figure 17: Rating Predictor Distribution. The red portion demonstrates the rating period of famous co-stars.

6.4 Movie Cast Rating

The movie cast rating model is evaluated using some great movie casts from the websites [10] and [9]. The ratings for some of these casts are provided in table 3. The percentile of their rating among a large number of casts from random movies is provided as well. As we can see, most of them have a high cast rating. The distribution of the mentioned random movie casts is shown in figure 18.

Movie	Cast Rating	Percentile
The Big Lebowski	0.647	93%
The Godfather	0.658	96%
12 Angry Men	0.597	55%
The Departed	0.661	96%
Inception	0.666	97%
Pulp Fiction	0.599	56%
American Hustle	0.696	99%

Table 3: Some Movies' Cast Ratings

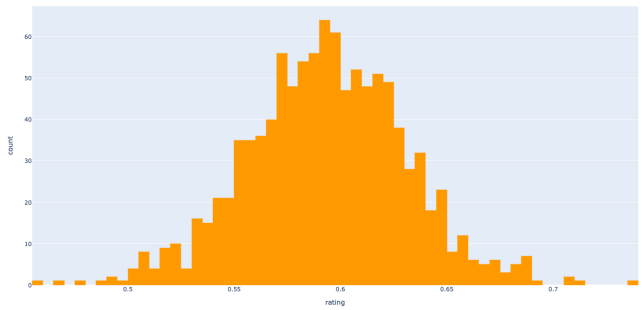

Figure 18: Movie cast rating distribution for 1000 random movies

7 CONCLUSIONS

Our experiments on the director-actor bipartite graph show that we cannot determine directors' characteristics only by the ratings of their common movies with actors. Moreover, the relation between an actor and a director depends on not only the rating of their common movies but also the genres of the movies, other cast and crews of the movies, and many other factors. If we want to predict a movie's rating, we cannot only consider the relation between its director and one of the actors. The success of a movie depends on the whole cast and crews, and all of them should be considered to generate features of a movie.

The movie similarity based alternative actor suggestion model is an excellent technique with our experiments. It shows us actors with lots of similar movies, and it provides us an intuition for its suggestion. Nevertheless, it is not practical because of slow queries; however, the second approach has better results according to the movie cast ratings and much faster running time.

Since the mean-squared-error on the link weight prediction task was close to zero, the trained graph autoencoder model leads us to high-quality results in the co-star suggestion and movie cast rating tasks.

REFERENCES

[1] O. Allali, C. Magnien, and M. Latapy. 2011. Link prediction in bipartite graphs using internal links and weighted projection. In *2011 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 936–941. https://doi.org/10.1109/INFCOMW.2011.5928947

[2] Rounak Banik. 2018. *The Movies Dataset.* https://www.kaggle.com/rounakbanik/the-movies-dataset

[3] BrightSide;. 2020. *15 Acting Duos Who Often Film Movies Together.* https://brightside.me/wonder-films/15-acting-duos-who-often-film-movies-together-and-were-happy-to-see-them-too-634760/

[4] Bernard Kurka. 2019. *Building a movie recommender system.* https://medium.com/@bkexcel2014/building-movie-recommender-systems-using-cosine-similarity-in-python-eff2d4e60d24

[5] Xin Li and Hsinchun Chen. 2009. Recommendation as Link Prediction: A Graph Kernel-Based Machine Learning Approach. In *Proceedings of the 9th ACM/IEEE-CS Joint Conference on Digital Libraries* (Austin, TX, USA) *(JCDL '09)*. Association for Computing Machinery, New York, NY, USA, 213–216. https://doi.org/10.1145/1555400.1555433

[6] Xin Li and Hsinchun Chen. 2013. Recommendation as link prediction in bipartite graphs: A graph kernel-based machine learning approach. *Decision Support Systems* 54, 2 (2013), 880–890. https://doi.org/10.1016/j.dss.2012.09.019

[7] Colin McCormick;. 2019. *Ranking The 10 Most Iconic Actor Duos.* https://screenrant.com/iconic-actor-duos-ranking/

[8] Andrew W. Moore;. 2017. *kd-Trees.* https://www.cs.cmu.edu/~ckingsf/bioinfo-lectures/kdtrees.pdf

[9] Brandon Sandberg;. 2018. *10 Great Movies With The Most Perfect Cast.* http://www.tasteofcinema.com/2018/10-great-movies-with-the-most-perfect-cast/

[10] TheTopTens;. 2021. *Greatest Movie Casts of All Time.* https://www.thetoptens.com/greatest-movie-casts-of-all-time/

[11] Phi Vu Tran. 2018. Learning to Make Predictions on Graphs with Autoencoders. In *5th IEEE International Conference on Data Science and Advanced Analytics.*