Movie Recommendations Based on Character Networks

Carah Alexander, Jorge Ochoa, Meeran Ismail

I. Introduction

The overwhelming number of movie options available instantly can make selecting a movie to watch feel like drinking from a firehose. Algorithmic movie recommendation systems help us cut through the sheer volume of options by filtering in only movies that the system "knows" we'd like. Many of these recommendation systems depend in large part on user-provided ratings for movies and recommend movies to a user based on what other users with similar tastes in movies have rated well. Thus we may not receive suggestions for less popular or well advertised movies if they don't have a critical mass of users that have provided rating information about them. This both limits a user's breadth of movie-watching experiences and can be harmful to small studios and independent filmmakers.
To combat this, we propose a content-based recommendation system that can surface and recommend these less well-known titles. We construct this system by extracting shared features from the character networks of movies.

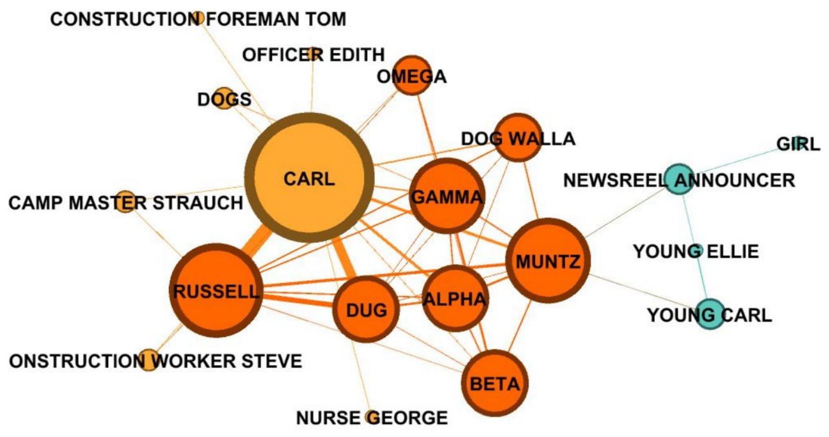


Figure 1: *The character network for the movie"Up"*

By relying on the character network of a movie, we can base our recommendations more on the narrative structure of movies, which can introduce more variation to a user's recommendations as well as give more exposure to smaller, independent films. In the process of creating a recommendation system, we also investigate which features of a movie's character network are most informative for predicting users' ratings for a movie and what qualities of these features make them relevant.

II. Relevant Prior Work

Mining and Modeling Character Networks

Bonato et al. looked at character networks from various novels, and analyzed several of those networks' properties such as diameter, clustering coefficient, and multiple centrality measures.

They found that using these properties of the networks, they were able to extract accurate literary conclusions and identify main characters as well as the communities of other characters around them. This suggests that we can likely depend on features of the character networks to make comparisons between movies and get meaningful results. Furthermore, the authors state that k-profiles by themselves were sufficient to distinguish between models when these were used as features for machine learning algorithm. This indicates that a handful of highly descriptive features could be enough to represent a given character network for the purposes of our research.

RoleNet: Movie Analysis from the Perspective of Social Networks
Weng et. al propose Social Network Analysis (SNA) as an alternative to audiovisual analysis for movies. They introduced RoleNet, a network where the more often two characters occur in a scene together, the more heavily weighted their edge and the "closer" those two characters are considered to be. Weng et al. used the centrality of nodes to determine the main characters of a movie. They also introduced algorithms for determining macro communities, communities including main roles, and micro communities, communities excluding main roles. Because they used the same metric (scene co-occurrence) to construct their social networks as *MovieGalaxies*, the algorithms proposed by Weng et. al in this paper would likely also be useful for the dataset we intend to analyze.

Learning Probabilistic User Models
Billsus and Pazzani looked at two different sets of text documents (academic grants and webpages), and then created user-specific content-based models that recommend documents based on a Naive Bayes Classifier that was trained on all movies rated by that given user, based on the 96 words with the highest expected information gain scores out of the corpus of documents that user had seen. With simple estimation bias and Laplace smoothing added on to these models, they were able to predict with close to 80% accuracy whether a user would find a document interesting. Because various network properties are analogous to words with high information gain, their results indicate that a regression analogue to the model used in the paper (e.g. linear regression, perhaps with regularization and constraints based on the minimum and maximum possible movie rating) using network properties as features could do well at predicting movies that users would rate highly.

Performance of Recommender Algorithms on Top-N Recommendation Tasks
Cremonesi et. al discuss various methods of solving the top-N recommendation task, which involves simply outputting the N movies that we think a user would like the most without trying to predict the exact rating that the user would give to those movies. Among other methods, the authors discuss the efficacy of neighborhood models, which recommend movies to users based on their being similar to movies that we know the user has liked. They found that these models performed fairly well, encouraging us to try a variant of this model ourselves.

III. Data

a. Datasets

MovieGalaxies Dataset

This dataset contains the character networks for over 700 movies along with various metadata about the movies. A node in the graph is a character in a movie. If two characters have been in at least one scene together, according to the movie's script, then there is an edge between them. The edge between two characters is weighted to reflect the number of times they've been in a scene together.

MovieLens 20M Dataset
This dataset contains 20 million ratings and 465,000 tag applications applied to 27,000 movies by 138,000 users; this is a subset of the available movie review data from the MovieLens website.

b. Data Processing

We had to make sure that we had a sufficient overlap in the data for the purposes of our project; effectively this meant only keeping ratings data for movies whose character networks we had and only keeping the character networks of movies for which we had ratings. We did this by cross checking against the movie's IMDB ids, which were present in both datasets, and eliminating all reviews that weren't for those films. 3% of the movies in our MovieGalaxies dataset did not have ratings in the MovieLens dataset, so we discarded those movies.

IV. Methodology/Algorithms used

Examination of the problem space
In our first attempt, for each movie's social network, we define the feature vector for a movie to be various attributes of its character network: its modularity, diameter, clustering coefficient, number of connected components, edge density, number of edges, and number of nodes. We built a linear regression model where the input was this feature vector for each movie and the output was the average rating of that movie, calculated as the average score given to that movie by all the reviewers in the MovieLens dataset who watched and reviewed that movie. Of the 749 movies in the MovieGalaxies dataset, we randomly chose 150 to set aside as a validation set, and then built the linear regression model using the remaining examples as training data.
Our next goal was to build a recommender system that could be tailored to individual users. Inspired by the work by Cremonesi et al. discussed above, we created a new feature space with only the most relevant features for each movie, and then ran a k-nearest-neighbors algorithm in this feature space to create a recommendation model.

Main and supporting character determination
We used the algorithms outlined in the RoleNet to separate leading from supporting characters and then to determine the micro and macro communities.
To isolate the main characters, we first calculate the centrality of each character as defined by RoleNet. The centrality of a given character is simply the sum of the weights of all their connected edges.
Once this is calculated, we then order the characters in descending order of centrality. We proceed through this list and calculate the difference in centrality between a character and its subsequent neighbor. The point in the list that has the greatest difference in centrality between

two characters marks the boundary between main and supporting characters. The character with the smaller centrality value, and all subsequent characters are supporting characters and the character with the larger centrality value and those previous are main characters.

Microcommunities
Microcommunities are communities comprised entirely of supporting characters. We determine them by first removing all main characters and any edges connected to them from the graph. We then sort all the remaining edges in the graph in descending order by weight. We initialize our communities by putting each supporting character in their own community. Then for each edge in our sorted edge list, if at least one character connected by this edge is in a community is of size one, we merge the communities containing the connected characters together.
The resulting communities are the micro communities for the given movie.

Macrocommunities
Given our set of microcommunities, we then reintroduce our main characters and their edges. We iterate through all edges connecting a main character to a supporting character. We assign each micro community to a main character based on which ever main character has the most weighty edge between them and some member of that community. The resulting set of all communities assigned to a given main character, including the main character themself, is a macrocommunity.

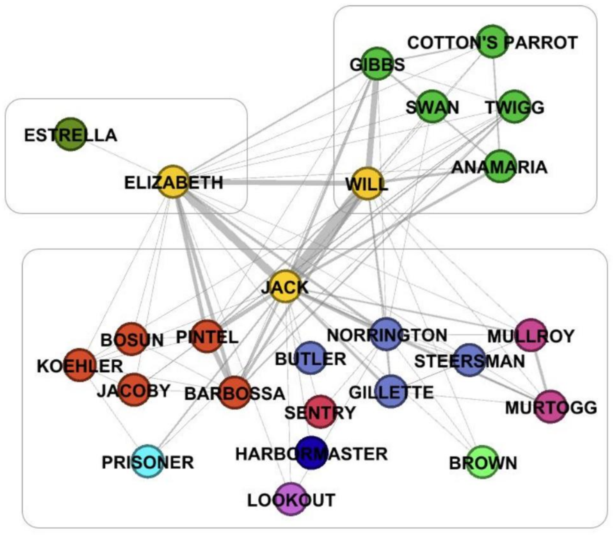


Figure 2: *The social network of Pirates of the Caribbean: The Curse of the Black Pearl. Nodes are color coded by microcommunity and boxed by macrocommunity.*

Motifs
We also compute the motifs in a movie's character network. To do this, we run the ESU Algorithm to compute the counts of all subgraphs of size 3 in a character network and compare to these counts to the motif/subgraph frequencies in our null model. We follow the same strategy as that in homework 2, question 3; namely, we use the configuration model as our null model, sampling it 10 times to get an estimated per-motif mean and standard deviation. We use a modified version of the code used in the homework to accomplish this, with the main changes

being made to the null model. While we originally performed 8000-10000 rewirings per graph in the homework, this many unique rewirings often are not possible on the character networks because they are of much smaller size. While the email and US power grid graph had on the order of thousands of nodes and tens of thousands of edges, character networks have at most about a hundred nodes and well under a thousand edges. Thus, we do only around a 100 rewirings for most graphs, and even less on graphs for which you cannot even do that many. The end result is an array of z-scores for each 3-motif per graph, and we use each z-score as another feature in our model.

Linear Regression

A linear regression model tries to model the relationship between the input features and the output as a linear relationship. In other words, this type of model assumes that output is just a linear combination of the inputs, i.e. the sum of the input features each multiplied by some weight. These weights (of which there is one per feature) are the parameters of the model that must be learned or estimated. In our case, we estimate the features using the least squares approximation. After these weights are learned, once can inspect them to determine the strength of the relationship between each feature and the output, i.e. whether there even exists a linear relationship between a given feature and the output in the first place. The strength of this linear relationship is determined by the magnitude of the weight corresponding to that feature.

V. Results

Linear Regression with basic features

With our initial approach, our model achieved a mean squared test error of .227 and a mean squared training error of .220. We observe from these results that our model was able to predict the average rating of a movie to within half a star.

Feature Weights

The linear regression model mentioned above produced the following weights after training:

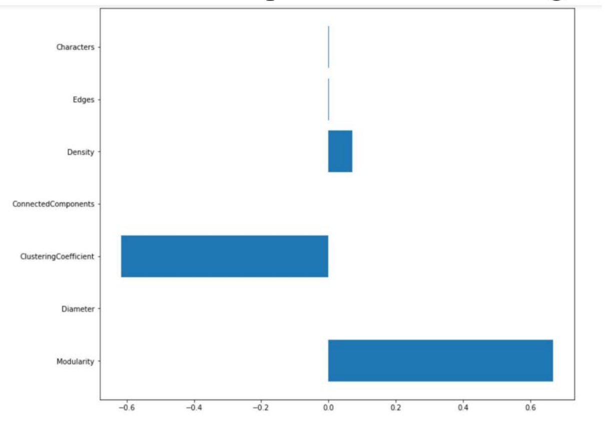


Figure 3: *The weights of features used in our initial linear regression model*

We noted that the most important determinants (from the features used in this model) of a highly rated movie are the average clustering coefficient of the nodes of the character network and the density of the character network. More specifically, the higher the clustering coefficient, the lower a movie's rating, but the higher the density, the higher the rating. We hypothesize that this is the case because a movie with too many detailed storylines / complex relationships may be too cognitively taxing to follow. Conversely, a densely connected character graph could be indicative of a movie that that is strongly centered around a few main characters, and thus has only a few important storylines throughout the movie. We explored this more by isolating main characters and looking at macro and micro communities in a network. Then, we created a linear regression model to predict a movie's average rating and obtained the following feature weights:

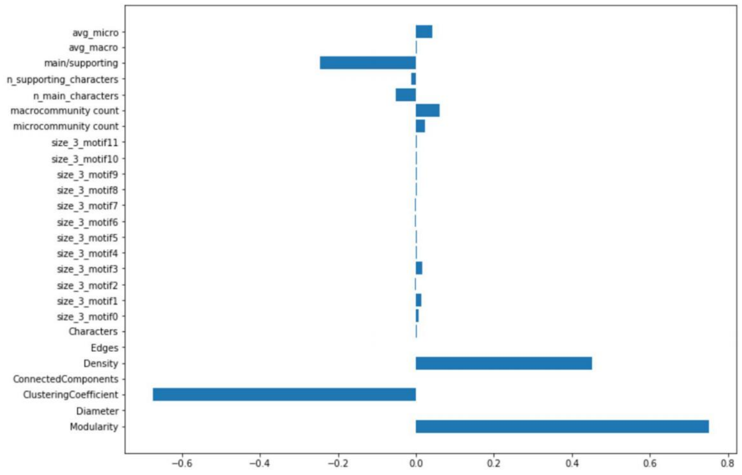


Figure 4: *The weights of features used in our initial linear regression model*

With both sets of features, the test set's mean squared error was 0.22; in other words, linear regression predicted ratings that were off by approximately about half a star. However, in both linear regression models, the predicted rating always fell between 3 and 3.7; in other words, linear regression just gives every movie a roughly average rating, which is why it's on average half a point off. From this, we see that linear regression based on the average ratings of the movies isn't predictively useful, which is why we focus on building a more personalized recommendation system.

While linear regression isn't useful for prediction, it does show which features are more strongly correlated with rating than others, and is a useful heuristic for choosing features. The heaviest feature weights occur primarily with certain graphwide features (density, clustering coefficient, and modularity) and the ratio of main characters to supporting characters. Motif features for the held little weight. Thus, given that those graphwide and community features are more correlated to average rating, we chose to use those features for our final model, a k-nearest-neighbors recommendation.

VI. Feature Analysis

The ratio of main to supporting characters was the only outstanding metric out of the character features, so we decided to examine this feature more thoroughly.

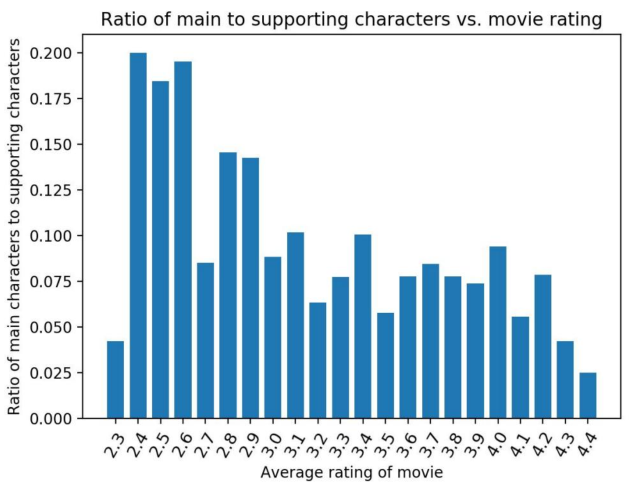


Figure 5: *Ratio of main characters to supporting characters vs average movie rating*

We noted that, for movies with average ratings of 3.0 or higher, rounded to the nearest tenth, the average ratio of main to supporting characters was consistently 0.1 or less. For average ratings less than 2.5, there were no more than 5 movies in each of those buckets so we excluded those when examining this trend. This could further suggest that viewers, on average, prefer to focus on a small number of main characters and their storylines, rather than on several equally important characters and narratives.

Community Feature Limitations:

In "Ferris Bueller's Day Off", our algorithm labels Ferris as a main character with a centrality of 60, while the next most "central" character, Cameron, has a centrality of 34. This is the biggest centrality gap, so Ferris ends up being marked by our algorithm as the only main character. The movie certainly centers around Ferris Bueller and is told from his perspective, so it is defensible to argue that he is the only main character. On the other hand, one could argue that Cameron is also a main character because he is so critical to the narrative. The first part of the film is dedicated to getting a depressed Cameron out of the house and all of Ferris' adventures in the city are with Cameron in tow.

We know that multiple main characters are identified when the story is told from their shared perspective, our algorithm considers both Harold and Kumar to be main characters in the film "Harold and Kumar Go to White Castle" and both Vivian and Edward to be main characters in "Pretty Woman". It is possible that when a movie is explicitly told from the perspective of one character, that character's necessary presence in each scene will result in a such a large character that presence of other key characters is drowned out.

We additionally realize that our main character extraction does not account for scenes where the character is in the scene alone. Since there are no self-edges, two characters must be in a scene together to create or further weight an edge between them. In "Ferris Bueller's Day Off", the movie begins with Ferris breaking the 4th wall and monologuing directly to the audience. Because he is not with anyone else in these scenes, they are not accounted for in our calculations, but they do add to his screen time and, thus to his status as a main character.

Motif Limitations:

We found that the weights assigned to motifs were among the lowest of all the features, indicating that their frequencies are not strongly correlated with the rating of a movie (at least not in any linear relationship). We believe that this is because our graph is undirected, so in reality there are only 2 unique 3-subgraphs possible; the z-scores that we computed assumed a directed graph and thus there was a lot of redundant information. The only information that can be extracted from the frequencies, namely the presence of either of the two possible undirected 3-subgraphs, is not useful because any given movie is likely to have both of them in its character network.

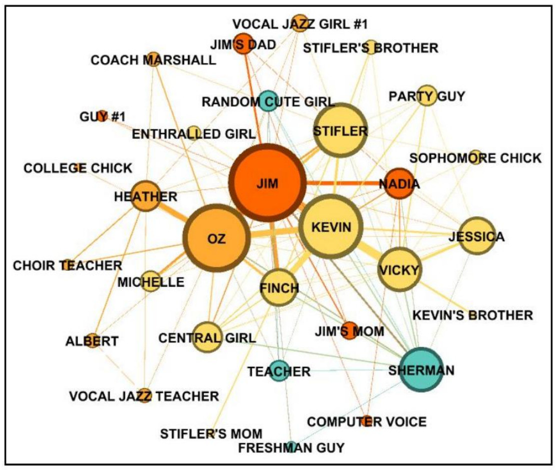
VII. Final Results

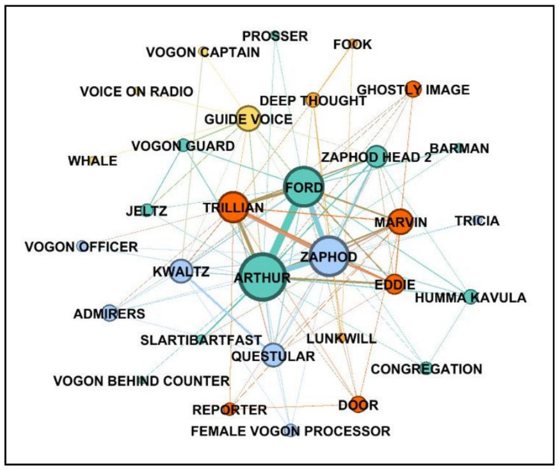
To evaluate the k-nearest-neighbors approach, we used the following evaluation metric:

1. For each queried movie, find the 4 nearest neighbors
    a. For each neighbor, find the set of users who've rated both the queried movie and the neighbor
        i. For each of these users, find the squared difference between their rating of the queried movie and their rating of the neighbor movie
        ii. Find the average of this difference across all users, across all neighbors for the queried movie; this is the queried movie's average squared loss
2. Take the average of the squared loss for all movies; this is the average squared loss of the k-nearest-neighbors approach

The intuition behind this evaluation metric is that if two movies are close "neighbors" in this feature space, then they are similar movies, and so if somebody watched both movies, that person should give those two movies similar scores.

On a k-nearest-neighbors approach using the relevant graphwide and community features, this evaluation metric gives us an average squared loss of 0.61, or an average loss of approximately 0.77. This means that our predicted ratings were wrong average by an average of 0.77 stars. This outperforms random selection, which has an average loss of 1.29 stars. While this appears to be worse than the linear regression model from earlier, because this approach deals with a specific queried movie, it has much less data to work with, and thus this result is still impressive.
As an example recommendation, querying *American Pie* on k-nearest-neighbors gives us *Charlie's Angels: Full Throttle, The Hitchhiker's Guide to the Galaxy, Pirates of the Caribbean: The Curse of the Black Pearl,* and *Cherry Falls*.





Features 6 and 7: *The character networks of American Pie (left) and The Hitchhiker's Guide to the Galaxy (right)*

Using random selection, alternatively, gives us *Ocean's Twelve, What Lies Beneath, Margot at the Wedding*, and *Remember Me.*

When considering the community features of American Pie and the Hitchhiker's Guide to the Galaxy we see the following relationship:

Feature	*American Pie*	*Hitchhiker's Guide to the Galaxy*	*Ocean's Twelve*
Main:Supporting	0.1	0.12	0.05
Macrocommunity count	3	3	2
Microcommunity count	13	10	20
Average macrocommunity size	11	10.3	21
Average microcommunity size	2	2	2
Number of main characters	3	3	2
Number of supporting characters	30	28	40

Here we see the similarities in the social structure of American Pie relative to the social structure of A Hitchhiker's Guide to the Galaxy.

Future Work

In future work, we would further refine our analysis of a movie's social structure to account for the feature limitations noted in this work. We would compute motifs for subgraphs of size 4 or greater. We would also accomodate scenes where only one character appears by adding self loops to make a character's centrality more accurate for the purposes of main character determination as described earlier.

VII. Conclusion

There appears to be a relationship between a movie's community structure and how a viewer would rate this movie. We know this because our k-nearest neighbors model outperforms our null model: randomly recommending k movies to the user. Using linear regression as a heuristic, we observed that the Density, Clustering Coefficient and Modularity of the social network graph, as well as the ratio of main characters to supporting characters are the most predictive features in determining a movie's rating.

We believe that we did equal work and would like to all receive the same grade.

Citations

Bonato A., D’Angelo D.R., Elenberg E.R., Gleich D.F., Hou Y. (2016) Mining and Modeling Character Networks. In: Bonato A., Graham F., Prałat P. (eds) Algorithms and Models for the Web Graph. WAW 2016. Lecture Notes in Computer Science, vol 10088. Springer, Cham

Cremonesi et. al. 2010. “Performance of Recommender Algorithms on Top-N Recommendation Tasks”. RecSys '10 Proceedings of the fourth ACM conference on Recommender systems, Pages 39-46

C. Weng, W. Chu and J. Wu, "RoleNet: Movie Analysis from the Perspective of Social Networks," in *IEEE Transactions on Multimedia*, vol. 11, no. 2, pp. 256-271, Feb. 2009.

F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. ACM Transactions on Interactive Intelligent Systems (TiiS) 5, 4, Article 19 (December 2015), 19 pages. DOI=http://dx.doi.org/10.1145/2827872

Kaminski, Jermain;Schober, Michael;Albaladejo, Raymond;Zastupailo, Oleksandr;Hidalgo, César, 2018, "Moviegalaxies - Social Networks in Movies", https://doi.org/10.7910/DVN/T4HBA3, Harvard Dataverse, V3

Github: https://github.com/jaochoa721/CS224W-Project