User Community Detection Based Recommendation Algorithm for MovieLens Database

Xiaoyao Wang, Lucia Zhang, Yutong Zheng

Washington University in St. Louis, St. Louis, 63130

December 12, 2019

Abstract

Movie recommendation systems are becoming more important with increasing amount of user information and greater demands of user experience. One of the basic ideas of recommendation system is based on user similarity - to recommend the preferred items of similar users. In this work we aim to design and evaluate a new recommendation algorithm by modifying the traditional collaborative filtering recommendation algorithm with community detection of users from a similarity network. We also aim to try different methods of creating the similarity network by using various measures of user similarity. The database used for this project is the MovieLens database downloaded from its official website[1]. We aim to predict the scores rated by users towards movies as the final output of our recommendation algorithm.

***Keywords*— Community Detection, Recommendation Algorithm, Similarity-based Weighted Network**

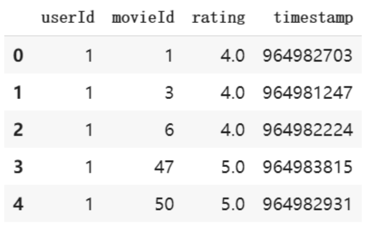
Introduction

A recommendation system is one class of information filtering system that seeks to predict the "rating" or "preference" a user would give to an item[2]. Recommendation systems are utilized in a variety of areas, and are most commonly recognized as playlist generators for video and music services like Netflix, YouTube and Spotify. One of the most traditional and wide-used technique used by recommendation systems is collaborative filtering, a method of making predictions about the interests of a user by collecting preferences information from many other users. The main goal of this project is to design and evaluate a new recommendation algorithm by modifying the collaborative filtering with community detection of users from a created similarity network.

There are mainly six stages of this project: 1) First we will conduct an exploratory data analysis and data wrangling to split the MovieLens database into training and testing sets. 2) Then we attempt to use the database, which offers information of ratings given by users towards movies and characteristics of movies, to create a baseline recommendation system using collaborative filtering technique. 3) For the next step, we will create a weighted similarity-based network of users from the bipartite network of users and movies. 4) We then plan to use one community detection algorithm for the weighted network to detect user communities and explore each community's preferred characteristics of movies as well as the similarities and differences among communities. 5) Based on the communities detected in the step four, we will modify the baseline recommendation system by combing the detected communities into the algorithm. 6) With evaluating the new system as the last step, we will discuss about the problems and possible further improvements of this project.

1 Exploratory Data Analysis and Data Wrangling

Considering the limitation of computation capability, we use the small dataset recommended for education and development by MovieLens. The dataset contains 100,000 ratings and 3,600 tag applications applied to 9,000 movies by 600 users, and was lately updated in September 2018. Below is a screenshot of the dataframe used to generate the baseline system and bipartite network (Figure 1).



	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931

Figure 1: Screenshot of ”ratings.csv”

By aggregately counting the ratings, we obtained the distribution of the number of users rated the movies verses the number of movies rated per user as in Figure 2. We filtered out users who rated below 20 movies and who rated more than 1,500 movies and had 100,836 records left. Then we randomly split the rating records of each user by 80-20 rule and used the 80 percent for training to make sure that one user appeared in both training and testing sets. Finally we got 80,419 records for training and 20,417 for testing.

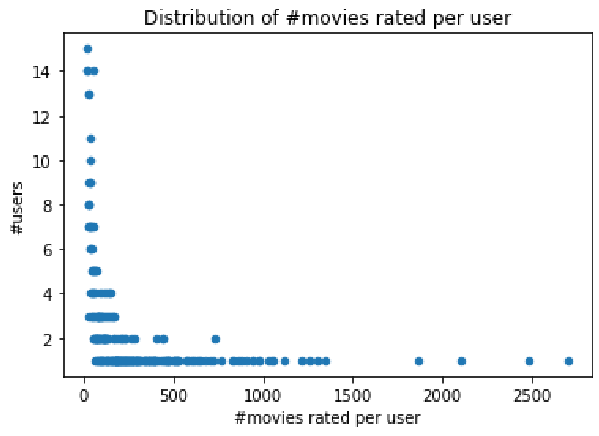


Figure 2: Distribution of number of movies rated per user

Besides the ”rating.csv”, the MovieLens dataset also provides information of genres for each movie, tags from users to each movie, and relevance score of each tag for each movie. Those data will be used in step 4 to detect patterns in communities detected.

2 Baseline recommendation system

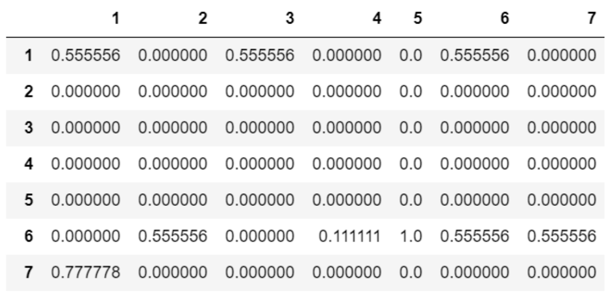
We used Collaborative Filtering[3], a commonly used basic method of offering simple recommendations, to build our baseline recommendation system. It is based on the assumption that users who agreed in the past will agree in the future, and that they will like similar kinds of movies as they liked in the past. Here the main idea is to measure similarity between users and recommend highly rated items (movies here) to our target users.

There are basically 5 steps that we did to build the baseline, stated as following.

2.1 Build Utility Matrix

The basis for conducting Collaborative Filtering is Utility Matrix, i.e. a matrix representing each users degree of utility for the items. In our Utility Matrix for movie recommendation, the rows are users and columns are movies, with each entry representing the users rating (normalized to scale of -1 to 1) for corresponding movies (Figure 3).

Since the range of utilities in the matrix is from -1 to 1, we can tell that a utility score close to 1 indicates that the user likes the movie, whereas the ones close to -1 show that the users dislike the movies. We used 0 to represent the ’blank’ entries in the matrix, where the rating information for the corresponding user and movie pair is missing. It is reasonable to do this since the best guess for a missing rating is that the user has a neutral attitude towards a movie he or she hasn’t watched yet.



	1	2	3	4	5	6	7
1	0.555556	0.000000	0.555556	0.000000	0.0	0.555556	0.000000
2	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000
3	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000
4	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000
5	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000
6	0.000000	0.555556	0.000000	0.111111	1.0	0.555556	0.555556
7	0.777778	0.000000	0.000000	0.000000	0.0	0.000000	0.000000

Figure 3: Utility Matrix

2.2 Compute User Similarity

We use utility vectors (rows in Utility Matrix) to compute similarity scores between user pairs. There are different ways to measure vector similarity, each with their own pros and cons. Here are some of the examples we have tried.

Cosine Distance[4]: the dot product of two vectors divided by the product of vector lengths. Higher cosine distance means the two vectors are more similar.

$$cosine\ distance = \frac{dot\ product}{product\ of\ vector\ lengths} \tag{1}$$

Jaccard Distance[4]: the number of commonly rated movies of two users (a and b) divided by the number of all movies rated by either a or b.

$$Jaccard\ distance = \frac{N(a) \cap N(b)}{N(a) \cup N(b)} \tag{2}$$

For these 2 similarity measurements, the Jaccard distance is more suitable for situations where the entries in Utility Matrix are binomial values (i.e. 0 and 1). In other words, Jaccard distance does not take into account the rating value, only considering the fact whether the user has rated the movie or not. Since the utility values here in our case are numerical instead of binomial, we eventually chose Cosine Distance as our similarity measurement.

2.3 Find n Most Similar Users

With the user similarity calculated, we can then rank the similarity scores of user pairs and find the most similar users for a certain target user. Here the idea is to use the most similar users as a representation of our target user, and We chose the 10 users with highest similarity scores for each target user.

2.4 Get Predicted Movie Rating

To predict our target user's potential rating for a certain movie, we calculated the average rating of these similar users who have seen this movie, and use this average number as our predicted value.

2.5 Compute Testing RMSE

Now that we have both predicted ratings and the actual ratings for the testing dataset, we can calculate the testing RMSE for our baseline recommendation system. Note that, to compute the RMSE, we need to rescale the normalized rating score (in range -1 to 1) back to the original scale of 0.5-5.

2.6 Result and Comments for Baseline Recommendation System

The resulting testing RMSE we got from our baseline recommendation system is 2.75, which is more than half of the range of the rating values (5-0.5), indicating that this approach is not valid.

To improve from our baseline, we tried two methods: one is to improve our similarity measurement, and another is to build network and detect communities from the user network. The following steps will discuss about theses two methods.

3 Create User Similarity Network from Bipartite Network

The original MovieLens dataset gives the ratings from users to movies, which leads to a bipartite network with two types of nodes - movies and users - and the ratings are the weights of edges in the bipartite graph. In order to do community detection, we first need to generate a similarity network of users from the bipartite one. To achieve this goal, we need to calculate the similarity between two users as the weight of an edge in the similarity network.

3.1 Calculation of Similarity Methodology

The similarity between two users are built on their similarity of ratings to the movies that they both viewed. There are three steps of calculation:

1. Normalize each rating record (from each user to each movie) to -1,0,or 1. The details of the calculation is:
   (a) For each user, calculate the median value of total ratings that he made;
   (b) compare the ratings from each person to the median value of that person we just found;
      i. If the rating = median, the score is normalized to be 0;
      ii. if the rating < median, the rating is normalized to be -1;
      iii. if the rating > median, the rating is normalized to be 1;
   (c) The -1, 0, and 1 here represent a positive, neutral, negative attitude from each user to each movie, respectively;
2. Calculate similarity between each pair of 2 users on each movie:

$$s_{m,12} = 2 - |r_{m1} - r_{m2}|, \tag{3}$$

3. Calculate similarity between each pair of 2 users on total of their commonly viewed movie:

$$W(u1, u2) = \sum_{m} s_{m,12} \tag{4}$$

3.2 Check Methodology Feasibility

This method only applies when the ratings of each movie are sparsely distributed, and the ratings from each person are also sparsely distributed, i.e. ratings of each movie do not center to a single score, and ratings from each person do not appear to be totally the same for all movies that he viewed.

3.3 Use Similarity Score as Weight of Edges to Build Network

We use the similarity score as weights of edges added to our new network of users. Below is a plot of the distribution of all edge weights.

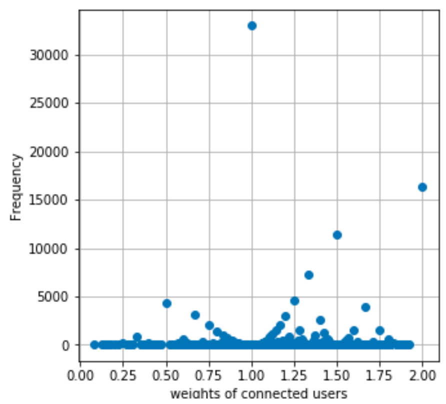


Figure 4: Normalized Similarity Score Distribution

From the distribution of the weights, we observed that most similarity scores (weights) are 1 (neutral) or 2 (highly similar), with floats distributed between them. However, not all weights are valuable to be add as edges. For those who have less weights (similarity score ¡=1), we can infer that those users are not similar.

After we tried several thresholds to filter the weights we want to keep, we found that keeping only the edges with weights equal 2 gives us the best partition in the process afterwards. Therefore, we add edges with weight ¿= 2 to our newly built graph as the similarity network of users.

3.4 Similarity Network Basic Information

The result similarity network has features as below:
Nodes: 609 (users)
Edges: 16360 (links if any pair of users are similar)
Average Degree: 53.727
Network Diameter: 4
Graph Density: 0.088
Connected Components: 1
Average Clustering Coefficient: 0.159
Average Path Length: 2.006

3.5 Similarity Network Evaluation Based on Basic Information

The evaluation to the similarity network is based on whether our network is representative of the network and could be used for partitioning.

Pros: The graph is fully connected and its density is 0.088, which is highly sparse for community detection.

Cons: The network diameter is 4, which may be too small for partitioning because nodes are not far enough.

Recommendation in the future: Apart from considering the similarity of the ratings, we can also consider other features such as the difference of the movies watched by a pair of users to delete some edges. As a result each user will be connected to less neighbors and the network can generate better partitioning results.

4 Community Detection and Patterns Discovered

4.1 Community Detection Algorithm

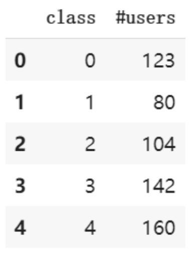
Having generated the similarity network from the bipartite network, We used modularity maximization algorithm, which was established during homework 5, to detect communities given a certain number of clusters. We also used Gephi(gephi.org), a software for network analysis, to obtain the best partitioning of the similarity network and Gephi used a technique called Louvain Method[5]. Both methods aim to maximize the modularity, a value between -1 and 1 that measures the density of links inside communities compared to links between communities[5]. For a weighted graph, modularity is defined as:

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j), \tag{5}$$

where A_{ij} represents the edge weight between nodes i and j; k_i and k_j are the sum of the weights of the edges attached to nodes$i$ and j, respectively; m is the sum of all of the edge weights in the graph; c_i and c_j are the communities of the nodes; and δ is Kronecker delta function $\delta_{x,y} = 1$ if $x = y$, 0 otherwise.

4.2 Patterns of Communities

After identifying the communities of users, we used other information from the MovieLens dataset, i.e. the genres and tag relevance of each movie, to analyze the communities. Though we tried several numbers of communities for our final recommendation system, we looked into the best partitioning one (Figure 6) to compare the similarity and difference among communities since it has the highest modularity. The best partitioning turned out to give five communities and the number of users in each community is given in Figure 5.



	class	#users
0	0	123
1	1	80
2	2	104
3	3	142
4	4	160

Figure 5: Number of users for each community/class

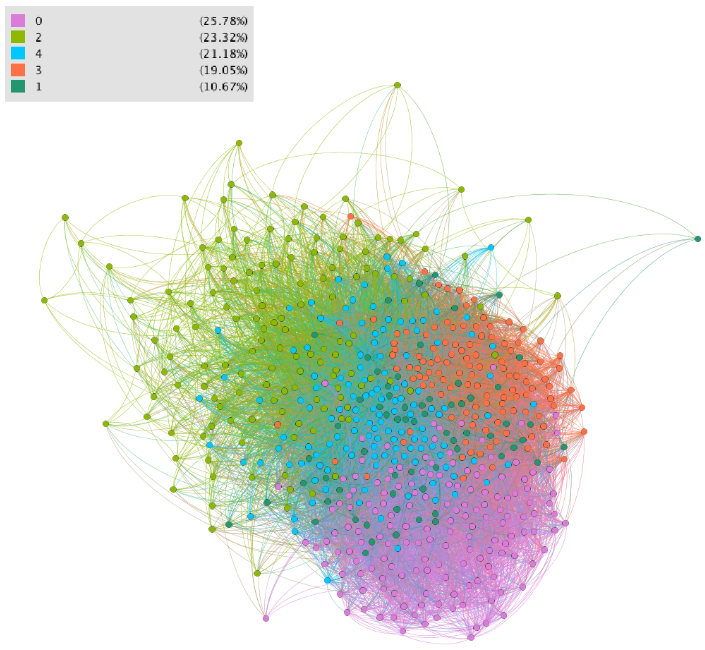


Figure 6: Community Best Partitioning in Gephi

Within each community, we then summarized the ratings for each genre by connecting the user-movie-rating file to the movie-genre file and obtained the ratings of genres for each community. The result was biased due to the different numbers of users per community and the different number of movies per genre, so we then normalized the result to remove these two effects and got the final result as shown in the heatmap.

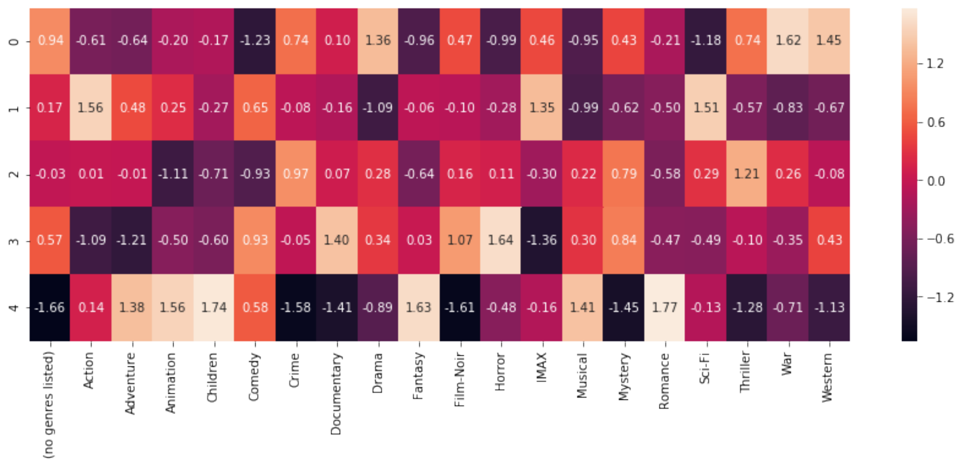
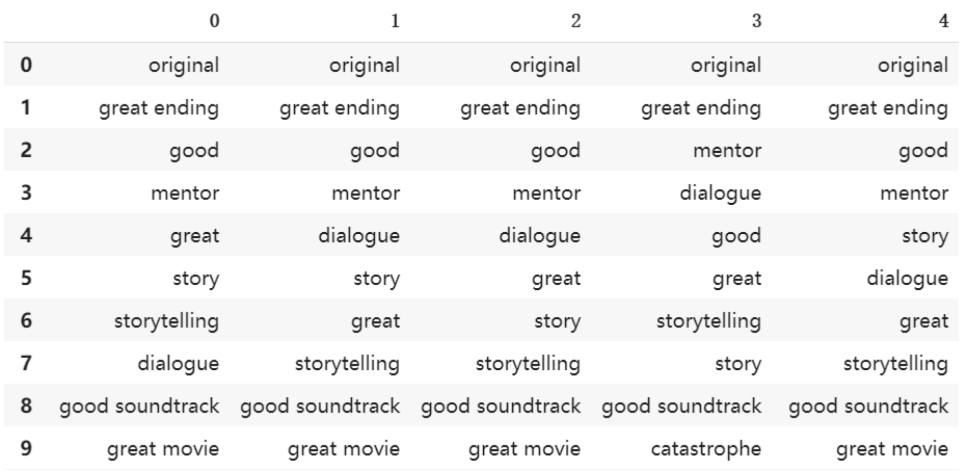


Figure 7: Genres rated for each community - Normalized version

As seen from the heatmap, the preferences towards genres vary among communities. Community 0 enjoys Drama, War, and Western, and dislikes Comedy and Sci-Fi. Users in Community 1 are fans of blockbusters and prefer Action, IMAX, and Sci-Fi to other genres. Kicksters from Community 2 love Crime and Thrillers while hate Animation. Community 3 chases after Horror, Documentary, and Film-Noir while despises Action, Adventure, and IMAX. It seems that this group enjoy thinking and have a sense of black humour. The users from last community are romantic persons. They love Romance, Fantasy, Children, Animation, and Adventure while hate Crime, Thriller, Documentary, Film-Noir, and Mystery. They have a preference towards beautiful things and reject anything frightening. Besides genres, we also summarized the relevance scores of tags and ranked the top 10 tags for each community as listed in the table below. The result shows that there is not much difference in terms of tags paid attention by users among communities.



	0	1	2	3	4
0	original	original	original	original	original
1	great ending	great ending	great ending	great ending	great ending
2	good	good	good	mentor	good
3	mentor	mentor	mentor	dialogue	mentor
4	great	dialogue	dialogue	good	story
5	story	story	great	great	dialogue
6	storytelling	great	story	storytelling	great
7	dialogue	storytelling	storytelling	story	storytelling
8	good soundtrack	good soundtrack	good soundtrack	good soundtrack	good soundtrack
9	great movie	great movie	great movie	catastrophe	great movie

Figure 8: Top 10 tags for each community

5 Community Detection Combined Recommendation System

5.1 Baseline with Modified Similarity Measure

The similarity measurement we used in our baseline recommendation system is Cosine Distance, and there are few drawbacks for this measurement. For instance, Cosine Distance tends to treat missing entries more like 'dislike the movie' than 'like the movie', which is not reasonable.

We tried a different similarity measurement method to improve from the result we got in our baseline recommendation system (described in section 2). That is the similarity measurement introduced in part 3.1. We believe that this measurement can solve the problem caused by Cosine Distance, thus providing more accurate similarity scores for user pairs and reducing testing error.

The new testing RMSE we obtained through the modified baseline recommendation system is 1.18, which shows a huge improvement (less than half) from the testing RMSE of our original baseline (which is 2.75).

5.2 Modified Baseline Within Community

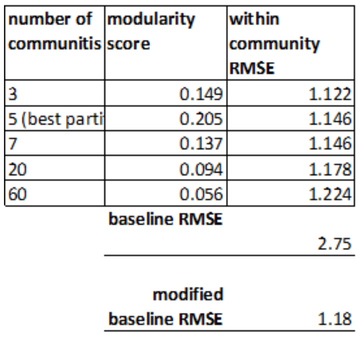
After modifying the baseline Collaborative Filtering, we then designed a new recommendation system by repeatedly running step 5.1 within each community detected in step 4. To be more specific, the list of neighbor users in order to recommend movies to candidate users is restricted to community, but not the whole set of users. A similar research was conducted by Gorripati and Vatsavayi in their work of a community based content recommender system[6]. A summarized process of the whole recommendation system is described below:

1. Split the training and testing datasets from the MovieLens database.
2. Based on the training set, build a similarity network of users from the bipartite user-rating-movie graph.
3. Detect n communities based on modularity maximization from the similarity network.
4. For the i-th community in n communities:
   (a) Pick out a set of records R_i from the training dataset where users U_i are in the i-th community;
   (b) Run the modified Collaborative Filtering on R_i to build the model M_i;
   (c) Pick out a set of records T_i from the testing dataset where users U_i are in the i-th community;
   (d) Predict ratings given userId and movieId in T_i using the model M_i.
5. Concatenate all the predicted ratings from n communities and compare them with the true ratings in the testing set to obtain the final evaluation measure, i.e. RMSE.

6 Evaluation and Recommendation

6.1 Evaluation of Model Results

Figure 9 is a summary of performance for various number of communities partitioned. We tried 3, 5 (best partitioning given by Gephi), 7, 20, and 60 communities and calculated the modularity of community detetcion and RMSE of the new recommendation system. One thing we can tell from the result is that our Collaborative Filtering Within Community Algorithm outpeforms the modified baseline in most cases, i.e. 3, 5, 7, and 20 communities, while results in a higher RMSE in the case of 60 communities. It seems that community detection does increase the accuracy of recommendation when the number of communities is small, and fails when the number of communities becomes large (60 communities vs. 600 users). Another thing notable is that higher modularity does not imply lower RMSE. In other words, a better partitioning with higher modularity does not necessarily cause a more accurate prediction of ratings.



number of communitis	modularity score	within community RMSE
3	0.149	1.122
5 (best parti	0.205	1.146
7	0.137	1.146
20	0.094	1.178
60	0.056	1.224
	baseline RMSE	2.75
	modified baseline RMSE	1.18

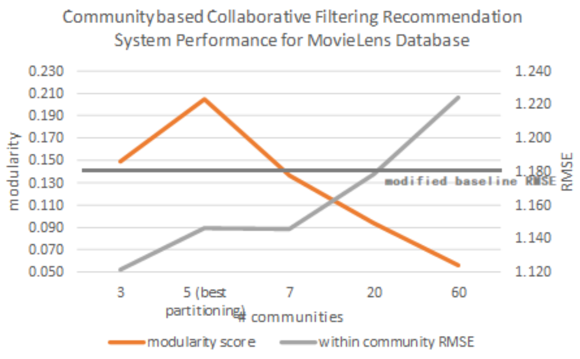


Figure 9: Community Based Collaborative Filtering Recommendation System Modularity vs. RMSE

6.2 Recommendation of Further Improvements

The minimum RMSE we got is 1.12 based on 3 communities, which left a great space for further improvements. In this section we bring up several methods that could possibly help create a more accurate recommendation system.

1. For Similarity Measure:
   (a) The similarity measurement we currently use for the utility matrix in Collaborative Filtering is gotten simply by adding the normalized rating (from -1 to 1) of 2 users together. In this way, we will actually give a higher weight for those user pairs who both give the movie a very high or low rating. That is, we will consider 2 users who both like or dislike a movie more similar than 2 users who both take a neutral attitude for a movie. This is a drawback of our current similarity measurement that we need to improve. Finding more appropriate similarity measurement, or adjusting based on this current similarity measurement, might give us more accurate similarity ratings thus improving testing RMSE.

   (b) The similarity measure used to build the similarity network has the same problem. It only takes into consideration the similar attitudes of each pair of users towards the same movie. Based on this measurement, we got many pairs of users that are fully similar (with similarity score equal 2). Therefore the degree of each node is too large for partitioning. In the next step, we can keep differentiating them by adding other features. Then by reducing the links among users based on their differences in other features, we can delete more edges to reduce the degree of each node, and generate a more sparse graph for partitioning. The potential features we could explore include the the diffencce between the number of movies rated by each and the total number of movies that they both reviewed, etc.

2. For Community Detection:
   (a) We used the modularity optimizaion to detect communities in this project, but there exist many other techniques such as spectral clustering and betweenness-based clustering.

   (b) In the real world, there could be overlapping among communities. One user can like blockbuster and also enjoy the crime movies at the same time. In this project, however, we only considered the non-overlapping case. We could further test techniques like clique percolation to detect the overlapping communities.

3. For System Algorithm:
   (a) We used Collaborative Filtering within communities detect for this project, but there are other applicable methods in terms of how to combine the community detection into the baseline. One is to utilize Collaborative Filtering between communities. That is to day, we could treat communities as 'super users' to reduce sparsity in the utility matrix.

4. Other Techniques:

(a) Besides RMSE, we could use other measures such as MAPE to evaluate the recommendation system.

(b) We could use cross validation and grid search in the training dataset to tune hyperparameters and thus to obtain a more accurate model. For instance, We understand that the number of community detected acts as an important hyperparameter in our recommendation system. As an improvement for our current system, we can pre-set different numbers of communities in our algorithm and compare the testing RMSE obtained from cross validation. In that way we can choose the best number of communities.

To conclude, in this project we designed a new recommendation system by applying collaborative filtering within communities detected from a similarity network generated based on a self-defined similarity measure. The results show that our algorithm does improve the accuracy of ratings predicted by the baseline system when the number of communities is not too large. We also listed several aspects where we can probably improve the performance of the models for further exploration.

References

[1] GroupLensResearch, "Movielens dataset," 2016.

[2] F. Ricci, L. Rokach, and B. Shapira, "Introduction to recommender systems handbook," *Recommender Systems Handbook*, pp. 1–35, 2011.

[3] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," *In Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, 1998.

[4] A. Rajaraman, J. Leskovec, and J. D. Ullman, "Section9.3 collaborative filtering," *Mining of Massive Datasets*, 2010.

[5] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, 2008.

[6] S. K. Gorripati and V. K. Vatsavayi, "A community based content recommender systems," *International Journal of Applied Engineering Research*, vol. 12, pp. 12989–12996, 2017.