Stanford CS 224W Project Final Report (Autumn 2015)
Title: Application of Network Analysis to Movie Recommendation

Jiang Han, Hanlu Huang, Chuiwen Ma
Stanford University, Department of Electrical Engineering

1 Introduction

This project is about recommendation system, a powerful and widely used technology nowadays, especially for Internet companies such as Google, Amazon, Netflix and etc. These companies own huge amount of user data, and they expect to extract additional value from the data, by building a recommendation system which can predict user's demands. Moreover, recommendation system can also improve the user experience of their platforms, because it helps user make decisions, thus saving their time. Generally speaking, recommendation system can recommend anything, such as music, movie, product, etc.

Companies value recommendation system since it brings additional income besides traditional promotion methods such as advertisements. In fact, a recommendation system can be interpreted as a personalized advertising engine. There are two important factors we need to consider for a recommendation system: performance and cost. In real applications, usually there are huge amount of user and item data. How to reduce the computational complexity becomes a very important issue. Also, since real data is often sparse, traditional solutions like collaborative filter (CF) will mostly not be able to predict all <User, Item> pairs, which yields bad system performance.

In this project, we propose a solution that is able to handle both of the two issues above, and implement our approach as a movie recommendation system. By applying graph generation and community detection algorithms to recommendation system, we are able to reduce the recommendation cost without sacrificing too much performance. Specifically, for single layer system, we can reduce the complexity to 3.67% of the original with only 2.8% loss of the overall Rooted Mean Square Error (RMSE). Further more, we propose a novel multi-layer recommendation engine. Each layer is utilizing a different information source such as movie vector, user vector, user attribute information and movie tags information. By doing this, we are able to exploit more useful information from the raw data, and are able to predict more cases, showing better performance than traditional approach. Notice that, the algorithm proposed in this report is not limited to a certain kind of dataset, but is extensible to any recommendation scenarios.

This report has four more sections. Sec.2 describes related works, Sec.3 gives a short analysis on the dataset, Sec.4 shows proposed methods and system implementations. Visualization and simulation results are shown in Sec.5. Sec.6 draws the conclusion.

2 Related Work

2.1 Community Detection Algorithms

Community detection is a core component of our recommendation system. With high quality community detection algorithm, we are able to significantly reduce the time complexity for prediction, and also reduce the noise in the data to some extent. There are many existing community detection algorithms. For example, a commonly used one is Girvan-Newman algorithm [1]. This algorithm finds the communities in a graph by removing the potential edges that connect different communities. It identifies those edges by computing the betweenness centrality of them. Edges with a high betweenness is likely to lie between two clusters of nodes. Although this algorithm gives reasonable quality results, it runs very slow on our dataset. The complexity of Girvan-Newman algorithm is $O(m^2 n)$, where n is the number of nodes and m is the number of edges.

Another popular community detection algorithm is Clauset-Newman-Moore algorithm [2]. This algorithm infers the community in a graph by greedily optimizing the modularity of the graph. By exploiting the shortcuts in the optimization, and using more complicated data structures, it runs a lot faster than Girvan-Newman algorithm. It's time complexity is $O(md \log n)$, where d is the depth of the dendrogram of the community structure. For a graph that has a hierarchical structure, $d \sim \log n$, so the running time is around $O(m \log^2 n)$. In this project, because our dataset is relatively large, we use Clauset-Newman-Moore algorithm instead of others.

2.2 Collaborative Filtering and Similarity Calculation

Collaborative filtering (CF) is an algorithm to compute a user's preference to an item, by collecting the preferences information from many users to many items. It is widely used in recommendation systems. Paper [3] gives a good summary of the CF algorithm. The first step of CF is similarity calculation. There are several kinds of metric we can choose from, such as cosine similarity, Pearson correlation coefficient similarity, and adjusted cosine similarity, etc. In this report, we compared these metrics and find the cosine similarity to be the best balance between performance and cost. The second step of CF is prediction. There are two kinds of prediction models: weighted sum and regression. The weighted sum approach computes the weighted sum of the raw user ratings, while the regression model uses an approximation of the raw ratings based on a regression model. For simplicity, in this project we use the weighted sum approach, because our goal is to show the power of networks, rather than use complicated non-network algorithms. In this project, we utilize the graph to generate communities, thus reduce the number of items needed to do prediction.

2.3 Principle of Multi-pass System

There are two important points for this project. One is applying graph community detection algorithms, described in Sec.2.1, to cluster those highly related users or movies, and consequently reduce the system runtime complexity. Another important principle we use is multi-pass system. This idea comes from Stanford multi-pass coreference resolution system [4].

Even though Stanford multi-pass coreference resolution system is originally used in natural language processing area, the principle of this paper could be extended to our recommendation system. Instead of the traditional single layer coreference resolution system, this paper applied multiple layers for coreference, and is expected to provide higher F1 score. Multiple layers are ordered from top to bottom, from highest to lowest precision, but from lowest to highest recall. By doing this, they can maximize the F1 performance of the system. In our recommendation engine, we are applying the similar layer ordering, i.e., we place layers with higher prediction precision on the upper layers so that they can make the predictions with higher priority. If the first layer can not predict the input <Movie, User> pairs, this pair will be further passed to the second or even lower layers for prediction. Those lower layers recommendation sub system may not have high precision as upper layers, but they may be able to handle additional data. Thus by doing this, we will be able to predict more input data with the most matched layer. Comparing with single layer recommendation, multi-pass recommendation system will have significantly better overall performance.

3 Data Collection

The data sets of this paper is collected by GroupLens Research from MovieLens web site. They are collected over time, including various amount of users, movies and ratings. The properties of different subsets are listed in Table. 1.

Dataset	Users	Movies	Ratings	Avg.User.Deg	Avg.Movie.Deg
100K	718	8,927	100,234	139.60	11.23
1M	6,040	3,883	1,000,209	165.60	257.59
10M	69,878	10,681	10,000,054	143.11	936.25
20M	138,493	27,278	20,000,263	144.41	733.20

Table 1: Dataset property

The users are randomly selected from a large database. All selected users have rated at least 20 movies. For the 100K, 10M, and 20M dataset, no user information except for user id is provided, while the 1M dataset also provides user's demographic information such as gender, age, occupation and zipcode. For movies, all of them have at least one rating or tag that are included in the dataset. Besides movie id, the title and genres are also provided. For ratings, each of them contains a user id, a movie id, and a 5-star rating score, with half-star increments (for 1M dataset, only whole-star ratings are included). Besides, we also have the tags with user id, movie id, and tag content, which provides additional information between user and movie.

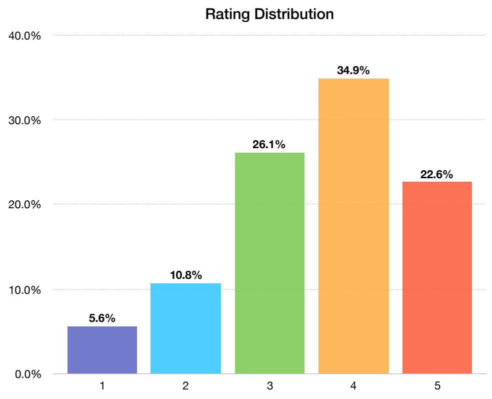
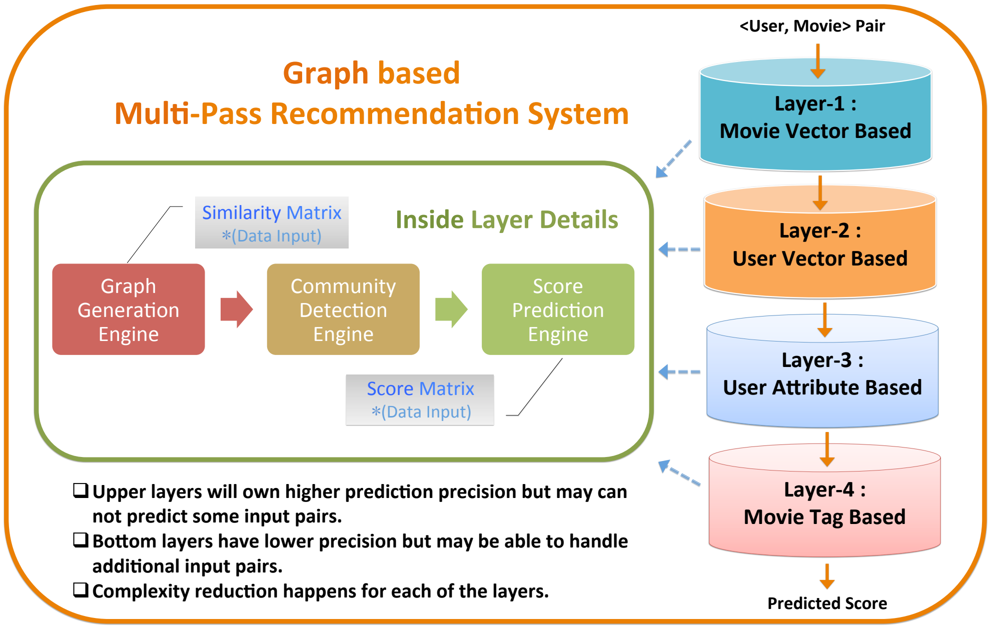


Figure 1: Rating distribution in 1M dataset

In this report, we mainly use the 1M dataset, because it has the highest $\frac{\#ratings}{\#users \times \#movies}$ score, i.e. the lowest sparsity. Another reason is that it contains user demographic information, which gives additional hint to build the connection between users. A drawback of this dataset is, it only includes whole-star ratings. Therefore, the prediction accuracy might be lower than using other datasets, because the input is more coarse-grained. Additionally, there is no tag information between users and movies, but this won't affect the elaboration of our algorithm. The rating distribution of 1M dataset is shown in Fig. 1. The average rating is 3.58, and the standard deviation is 1.117.

To split the dataset into training set and test set, we randomly select 10,230 ratings as the test data, and take the rest as the training data. The training data is used to compute similarity matrices, conduct community detection, and generate rating predictions.

4 Methods and Implementations



<User, Movie> Pair
Graph based
Multi-Pass Recommendation System
Layer-1 : Movie Vector Based
Layer-2 : User Vector Based
Layer-3 : User Attribute Based
Layer-4 : Movie Tag Based
Predicted Score
Similarity Matrix *(Data Input)
Inside Layer Details
Graph Generation Engine
Community Detection Engine
Score Prediction Engine
Score Matrix *(Data Input)
❑ Upper layers will own higher prediction precision but may can not predict some input pairs.
❑ Bottom layers have lower precision but may be able to handle additional input pairs.
❑ Complexity reduction happens for each of the layers.

Figure 2: Graph based multi-pass recommendation system

In this part, we introduce the design principle and details of the graph based multi-pass recommendation system, shown in Fig.2. In this design, there are four layers in the entire system. Each layer has a similar framework, shown in the box "Inside Layer Details" in Fig.2. The two subsections below introduce each layer's framework and the overall multi-pass system respectively. In general, the graph based design for each layer will be able significantly reduce the prediction complexity without losing too much RMSE performance. Hence each layer will provide reasonable prediction result with lower complexity. Since each layer uses different information to generate graph and has different similarity matrix metrics, their predictable <User, Movie> pairs are different. Ideally, with multi-layer system, we should be able to handle more input <User, Movie> pairs, thus has higher system recall.

4.1 Inside Layer Detail Introduction

4.1.1 Graph Generation Engine

In this part, we are interested in generating corresponding graphs based on the "Similarity Matrix" (as shown in Fig.2). Similarity Matrix can be either "user-to-user" similarity or "movie-to-movie" similarity, and will generate user graph or movie graph.

We compressed the network using one-mode projection onto user graph and movie graph. To do this, we firstly calculate the similarity between every pair of users and every pair of movies. Each pair of nodes in the projection network is connected only if its similarity is larger than a threshold. The principle is that we can filter out low similarity edges and in turn reduce the complexity without sacrificing too much performance. Similarity calculation depends on what information we have for either movie or user. The main difference of different layers is the similarity calculation metric and the information applied, which will be explained in Sec.4.2.

4.1.2 Community Detection Engine

The graph generation step will output two graphs: User graph and Movie graph. After the filtering step, the remaining edges indicate that the two connected nodes have a reasonably high similarity, thus is supposed to contribute more to the prediction. Furthermore, if we use all user or movie nodes for prediction, the run time complexity will be very high. Therefore, we will use community detection approach to cluster both user and movie nodes. By doing this, similar nodes will be grouped into the same community, and we are able to do prediction only base on the nodes in the same community with relatively lower complexity. We are interested to compare the results of complexity and RMSE performance, we are looking forward to finding the parameters that provide near benchmark performance with lower complexity.

In the current system, we are using Clauset-Newman-Moore community detection algorithm, an algorithm suitable for large networks. Table 2 shows the detected community numbers of both movie and user graphs with different thresholds. From which we can see that with higher threshold, we will detect more communities, because the graph will become sparser.

Threshold	0.05	0.1	0.15	0.2	0.25
User Community	6	43	231	456	576
Threshold	0.3	0.5	0.6	0.7	0.8
Movie Community	8	16	29	40	55

Table 2: Detected community number for movie and user graph

4.1.3 Prediction Engine

In this step, we want to predict the rating of a particular user to a particular movie, which can also be seen as weighted link prediction problem in the original bipartite graph.

The members in one community are usually similar to each other. To predict a particular rating, we could make use of the ratings from the members in the same community. Specifically, we use the following formulation to calculate the prediction:

$$\hat{r}_{xi} = \frac{\sum_{j \in N(i;x)} S_{ij} \cdot r_{xj}}{\sum_{j \in N(j;x)} S_{ij}}$$

In this step, our inputs are two sets of communities: user communities and movie communities. For each kind of communities, we can build a prediction engine. To interpret the meaning of the results, assuming we are using the user graph, the prediction process considers the similarities between the input user and the other users in the community where this user belongs to, as well as the edges between those users and the input movie.

4.2 Multi-pass System Introduction

This section will introduce how the multi-pass system is organized and the detail for each layer. As mentioned, multi-pass system is able to predict more <user, movie> pairs, thus increases system recall. This system has the following characteristics:

- Upper layers have higher prediction accuracy but may not be able to predict some input pairs. The principle here is that we want to apply those layers with better prediction performance first on the input <user, movie> pair, so that we can get a lower overall RMSE.
- Bottom layers have lower precision but may be able to handle additional input pairs. If a specific <user, movie> pair cannot be handled by the upper layers, then this pair will be passed down to bottom layers, until it is predicted or even the last layer cannot predict it.
- Complexity reduction happens in each layer. Each layer uses the same framework, thus threshold tuning and community detection is used for each layer with reduced complexity. Although we have four layers in the system, but only one of the layer will be used to run the actual prediction for the input. Therefore, multi-layer system is not increasing the overall complexity but can provide better recall performance.

Each layer has different predictable set since they are using different information to generate the similarity matrix, which will generate different graphs and community topology. In the following parts, we will introduce the detail for each layer.

4.2.1 Layer 1: Movie Vector Based

For layer 1, we are using the movie vector to represent each of the users, since in the experiment, it performs better than other methods, i.e. every user is represented as a vector of the movie ratings he/she rated (rating will be 0 if this user did not rate the movie). Therefore every user will be represented by the same dimensional movie vector. Then we can populate the user similarity matrix by calculating the cosine similarity of any two users. Similarity matrix will be symmetric.

The function to calculate cosine similarity is:

$$sim(x, y) = \arccos(\mathbf{r}_x, \mathbf{r}_y) = \frac{\mathbf{r}_x \cdot \mathbf{r}_y}{\parallel \mathbf{r}_x \parallel \cdot \parallel \mathbf{r}_y \parallel}$$

However, in cosine similarity, it treats missing ratings as "negative", which is not very reasonable, because if a user haven't seen a movie, it doesn't necessarily mean the user doesn't like the movie. In order to solve the problem we centered the data at 0, which is to subtract the ratings by the average rating of a user or a movie.

4.2.2 Layer 2: User Vector Based

For layer 2, we are using user vector generated graph. Here, every movie is represented as a vector of ratings from all the users (rating will be 0 if this user did not rate the movie). Thus each movie will have the user vector of same dimension, which equals to total user number. Then we can populate the Similarity Matrix by calculating the cosine similarity of any two movies. We also tried Pearson Correlation Coefficient similarity, but cosine similarity is chosen after the performance comparison. Therefore, layer 2 will generate the movie graph and apply community detection, and following prediction based on this graph.

4.2.3 Layer 3: User Attribute Based

For layer 3, we apply the user attribute information, as shown in Table 3 below:

UserID	Gender	Age	Occupation	Zip-code
1	F	18	10	48067
2	M	56	16	70072
3	M	25	15	55117
4	M	45	7	02460
5	M	25	20	55455
...	...	...	...	...

Table 3: User attribute information in the data

From which we can see that each user has four features here, i.e. Gender, Age, Occupation and Zip-code. For age and occupation, different number indicates different range or category.

For gender, we have: "F: female", "M: male".

For age, we have: "1: Under 18", "18: 18-24", "25: 25-34", "35: 35-44", "45, 45-49", "50: 50-55", "56: 56+".

For occupation, we have: "0: other or not specified", "1: academic/educator", "2: artist", "3: clerical/admin", "4: college/grad student", "5: customer service", "6: doctor/health care", "7: executive/managerial", ...

Hence, we are able to apply the above user attributes to evaluate two user's similarity. Here we apply the principle of linear regression in machine learning:

$$userSim = w_1 f_1 + w_2 f_2 + w_3 f_3 + w_4 f_4$$

Here, let's define the four features: Gender as f_1, Age as f_2, Occupation as f_3 and Zip-code as f_4. Where f_i will be 1 if the two users have same corresponding feature category, otherwise will be 0. And we have constraint $\sum_{i=1}^{4} w_i = 1$ for the four weights. We tune the weights to give the best single layer performance. Basically, weights here indicate the importance of the corresponding feature on deciding user's rating on the specific movie. Now we are able to populate the similarity matrix by calculating any two user's similarity based on their four above features.

4.2.4 Layer 4: Movie Tag Based

In this layer, we use the given the movie tag information such as following:

Movie Name	Tag Information
Toy Story	Animation, Children's, Comedy
Jumanji	Adventure, Children's, Fantasy
Cutthroat Island	Action, Adventure, Romance
Get Shorty	Action, Comedy, Drama
... ...	....., ...

Table 4: Movie tag information in the data

Therefore, we can apply the information in above table to calculate the similarity of movies. Here, we use the basic Bag-of-words model from natural language processing. Specifically, each of the movie will have one bag carrying all of its tags. For example, movie "Toy Story" in Table 4 will have bag with {animation, children's, comedy} (all the tags are transformed to lower case before going into the bag). And movie "Jumanji" will have bag with {adventure, children's, fantasy}. Based on the tag bags, we use Jaccard similarity to indicate the similarities of any two movies. Jaccard similarity formula is shown as following:

$$JaccSim = \frac{|A \cap B|}{|A \cup B|}$$

That is, Jaccard similarity of A and B is the division of A, B's intersection size and A, B's union size. Therefore, movie "Toy Story" and movie "Jumanji" in Table 4 will have Jaccard similarity of 0.2. Hence in layer 3, we are able to populate the similarity matrix by calculating the Jaccard similarity based on the tag bags of any two movies. Thus being able to provide different movie graph and prediction results.

5 Simulation Results and Analysis

5.1 Evaluation Metrics

To show a complete performance of our proposed method, we have three evaluation metrics for this project, which are: RMSE to evaluate the prediction precision, prediction cost to evaluate the running complexity of the algorithm and predictable ranking to evaluate how many <User, Movie> pairs our method could predict (this metric is similar to recall).

5.1.1 RMSE

We use RMSE to formulate the difference between our predicting results to the real rating values. In the following expression, $\hat{r}$ is our predicted rating, r is the ground truth rating. n is the number of test data. The lower is RMSE value, the closer is our prediction to the real ranking score.

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n} (\hat{r} - r)^2}{n}}$$

5.1.2 Prediction Cost

One innovative point of this project is to apply graph community detection to reduce the prediction complexity by only using neighbors in the same community where input user or movie belongs to. Here, we use prediction cost to reflect the prediction complexity. It equals to the average number of neighbors used when doing prediction. Smaller prediction cost indicates lower system complexity.

5.1.3 Predictable Rankings

In community detection engine, those edges with weights smaller than a specific threshold are omitted to reduce complexity. This could generate some unpredictable nodes, especially for those nodes less similar to others. However, this graph pruning will not influence the performance too much because those edges are pruned since they have very low similarity value. Here is when multi-pass system comes into working. Since different layers have different similarity calculation metric (1-st layer: user vector, 2-nd layer: movie vector, 3-rd layer: user attributes, 4-th layer:

movie tag), those nodes can not be predicted on upper layer will very likely to be predicted by lower layers. Ideally, an recommendation system should be able to predict all the cases no matter how many movies the user has watched or how many users has rated the movie. Thus here, we define the number of nodes that the system can predict as another evaluation metric (similar to recall).

5.2 Graph Generation Engine Performance (Degree Distribution)

The degree distribution of the user-user graph and movie-movie graph are shown in Figure 3, which follows power law distribution.

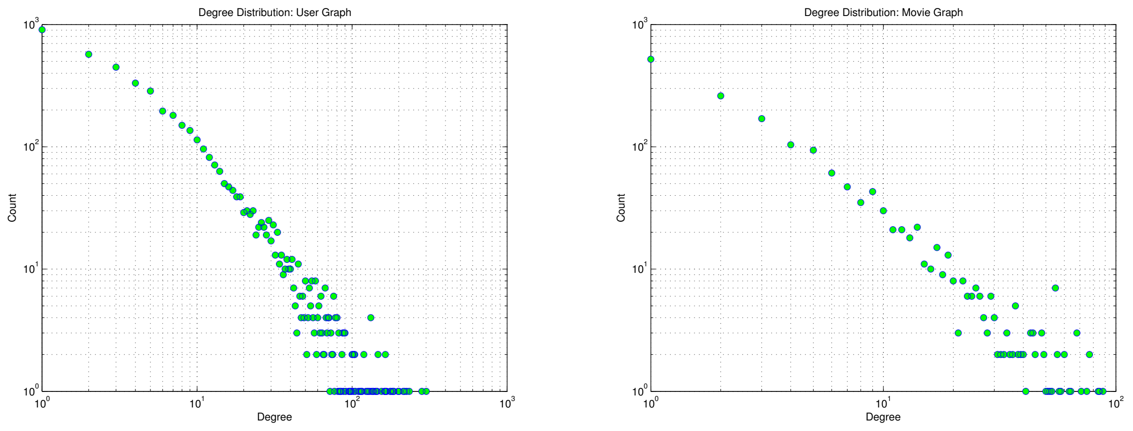


Figure 3: User graph degree distribution (left). Movie graph degree distribution (right).

5.3 Community Detection Engine Performance (Visualization)

Fig. 4 shows the visualization result of user graph using Gephi. Note that this figure is just a subsection of the total user nodes, other parts are filtered in Gephi for better visualization display.

5.4 Prediction Engine and Multi-pass Performance

5.4.1 Threshold Tuning

In the first step of graph generation, we need to set a threshold to filter out edges whose weights (which are also the similarities between nodes) are smaller than this specific threshold. By using different threshold values, we would get different graphs and different performances (tradeoff of precision and complexity). Taking user vector layer as example, table 5 shows how threshold would affect prediction performance.

Here, benchmark is calculated using all the valid nodes in the whole graph instead of in the community. As we can see, when increasing threshold, prediction RMSE value is also increasing, with decreasing predication cost and predictable rankings. It is because larger threshold will parse out those low similarity edges, generating more sparse graph and smaller communities. However, since the detected community only includes members with very high similarities, higher threshold will not hurt RMSE too much. Table 5 shows that when we take threshold as 0.25, the prediction

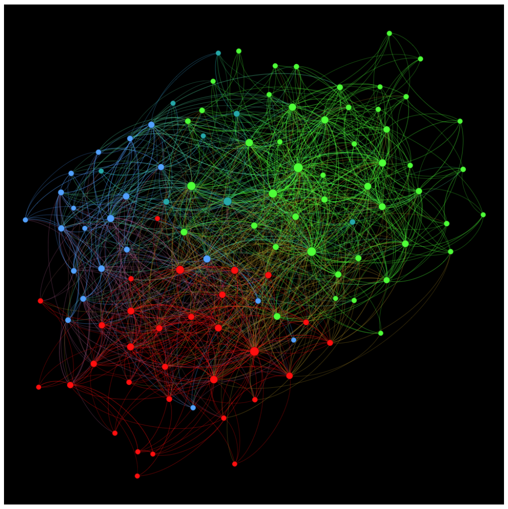


Figure 4: User graph community detection visualization (part nodes are filtered for better display)

Threshold	0	0.05	0.1	0.15	0.2	0.25	0.3
Prediction RMSE	0.9017	0.9059	0.9131	0.9156	0.9171	0.9194	0.9810
Benchmark RMSE	0.9008	0.9031	0.9052	0.9052	0.8974	0.8943	0.9005
Prediction Cost	172.8	171.0	163.8	91.9	59.3	11.7	2.3
Benchmark Cost	299.0	299.1	299.7	305.4	315.2	318.2	323.4
Predictable rankings	10230	10228	10188	8973	5027	2696	1344

Table 5: Prediction performances using user vector with respect to different thresholds

cost will be reduced from 318.2 to 11.7 (3.67%), while the RMSE performance loss is only from 0.8943 to 0.9194 (2.8%).

The performance in other layers share similar features. Our goal of tuning thresholds is to achieve relatively low RMSE with low cost but more predictable rankings.

5.4.2 Layout Arrangement

In order to be able to predict all the rankings using our methods, we used four prediction layers in our project. To arrange the layers in an appropriate sequence, we test the performance of every single layer individually. The result is shown in table 6.

Layer	movie vector	user vector	user attributes	movie tag
Threshold	0.15	0.15	0.5	0.15
Prediction RMSE	0.8883	0.9156	0.9877	1.0047
Prediction Cost	335.6	91.9	183.9	107.98
Predictable rankings	9834	8973	10217	10207

Table 6: Prediction performances using single layer performance with different similarity metrics.

The threshold above is chosen by doing trade off among the three evaluation metrics. To achieve the best system performance, we rank the layers based on their precision (RMSE) so that we can guarantee the <User, Movie> pairs be predicted with the most accurate score. Hence, we have: movie vector > user vector > user attributes > movie tag. Therefore, we set movie vector as the first layer, user vector as the second layer, user attributes as the third layer and movie tag as the fourth layer.

The result meets our expectation that the performances of user and movie graphs are obviously better than graphs generated by user attributes and movie tags. Because the first two graphs contain the ranking information but the last two graphs only use the attributes of users and movies, which are more like content-based recommendation system. Also, even though that prediction cost is different among the four layers, they are all reducing the complexity significantly comparing to the benchmark while with smaller RMSE loss.

5.4.3 System Performance

Combining the four layers as the complete multi-pass system, we can get the performance of the whole recommendation engine, which is shown in table 7. The thresholds used in the final system are also tuned a little bit to better fit in the multi-pass system. To choose the thresholds in the whole system, the parameters in the previous layers do not necessarily need to be able to predict a lot of rankings, but put more emphasize on reducing costs.

Layer	movie vector	user vector	user attributes	movie tag	overall
Threshold	0.25	0.15	0.4	0	/
Prediction RMSE	0.8926	0.9318	1.0469	0.9891	0.9279
Benchmark RMSE	0.8602	0.9070	1.0423	0.9891	0.9031
Prediction Cost	99.3	75.7	124.1	134.6	88.1
Benchmark Cost	807.3	263.1	448.2	134.6	437.4
Predictable rankings	3237	6053	451	489	10230

Table 7: Prediction performances using multi-pass system

The RMSE, prediction cost and predictable rankings in table 7 are all corresponding to the specific layer. That is, the second layer will only work on data can not be predicted by first layer, third layer will only work on data not predicted by second layer, etc. This is why we see some difference of Table 7 and Table 6 (where each layer is runing separately). In all, every layer in the multi-pass system is responsible for a portion of the ranking results.

Regarding the prediction cost, we can see that each layer (except last layer) is reducing the cost comparing to benchmark significantly. Last layer is assigned with threshold of 0 since we want the last layer to handle all the remaing <User, Movie> pairs. Even though the last layer will own same cost as benchmark, while the remaing pairs are just 489 (4.78% of total data), thus this will not influence the overall complexity. As we can see, multi-pass system overall cost is 88.1, which is only 20.1% of the benchmark. But the overall multi-pass prediction RMSE precision is 0.9279, which loses only 2.75% comparing to benchmark. Furthermore, The whole system is able to predict all test cases (the number of test cases is exactly 10230 rankings). But if we only apply the first movie vector layer, we are only able to predict 3237 (31.6%) samples. Hence, the multi-pass system is able to reduce complexity significantly while loses little RMSE performance. Meanwhile, it owns significantly higher recall comparing to single layer system.

6 Conclusions

In this project, we proposed a graph based multi-pass recommendation system. We show the design of each layers, which applies information like user vector, movie vector, movie tags, user attributes. We also show the details inside the layer framework, including graph generation engine, community detection engine and prediction engine. The proposed system is able to reduce the prediction complexity significantly while without losing too much RMSE performance. Also, it will be able to predict more input <User, Movie> pairs than single layer system and thus having better overall performance.

References

[1] Girvan M, Newman M E J. Community structure in social and biological networks[J]. Proceedings of the national academy of sciences, 2002, 99(12): 7821-7826.

[2] Clauset, Aaron, Mark EJ Newman, and Cristopher Moore. "Finding community structure in very large networks." Physical review E 70.6 (2004): 066111.

[3] Sarwar, Badrul, et al. "Item-based collaborative filtering recommendation algorithms." Proceedings of the 10th international conference on World Wide Web. ACM, 2001.

[4] Lee, Heeyoung, et al. "Stanford's multi-pass sieve coreference resolution system at the CoNLL-2011 shared task." Proceedings of the Fifteenth Conference on Computational Natural Language Learning: Shared Task. Association for Computational Linguistics, 2011.