IEOR 4571
Personalization: Theory and Application
Project 1 Report

Tao Li (tl2863)
Naili Ding (nd2588)
Chaoyue Zheng (cz2529)
Jinglin Chen (jc5059)

11/06/2019

1. Abstract

The team aims to develop a recommendation system for a digital media company that needs to recommend movies. Two collaborative filtering algorithms have been developed: item-based collaborative filtering and matrix factorization to recommend movies to users based on movie rating data.

2. Objective

The objective of the project is to build a recommendation system for a digital media company. The recommendation system is designed to recommend correct and various movies to users which can improve user experience, attract new users and retain current users. In this case, the digital media company can be more professional and popular.

The team is trying to optimize RMSE and MAE of models to get as accurate recommendations for users as possible. Due to the complexity of the item-based collaborative filtering model, the team only trained a subset (movie ratings created within the most recent two years) of the whole dataset for the models. The recommendation system is built to serve for users, which can provide accurate movie recommendations for them. In this case, to maximize user satisfaction, it is important to develop a model with low RMSE and MAE. Moreover, to provide users with various recommendations rather than limited number of movies, the model needs to have large coverage value.

3. Data and Methods

3.1. Initial Data Selection

The raw dataset used contains 20000263 ratings that is created by 138493 users across 27278 movies between January 09, 1995 and March 31, 2015. The team considers that users' preference of movies might change over time and using the whole dataset seems to be imprudent. Since the raw dataset contains all the ratings for over 20 years, it may not be able to represent users' current tastes. Due to the complexity of the item-based collaborative filtering algorithm and the calculation power of the machine, the team was not able to train a large amount of data over the models.

Therefore, ratings that were created in the most recent two years (from March 31, 2013 to March 31, 2015) were selected to develop the model because the amount of the subset is manageable and these data can represent users' current preference well to help the recommendation system performs better. The subset contains 1281535 ratings that came from 13092 users for 22648 movies. As we can see from Figure 1, most of the movies have a rating of 4. From Figure 2, we can see that the distribution of number of ratings per movie has a long tail on the right side. Also Figure 3 indicates the top 10 rated movies from the subset. Figure 4 shows the distribution of number of ratings per user.

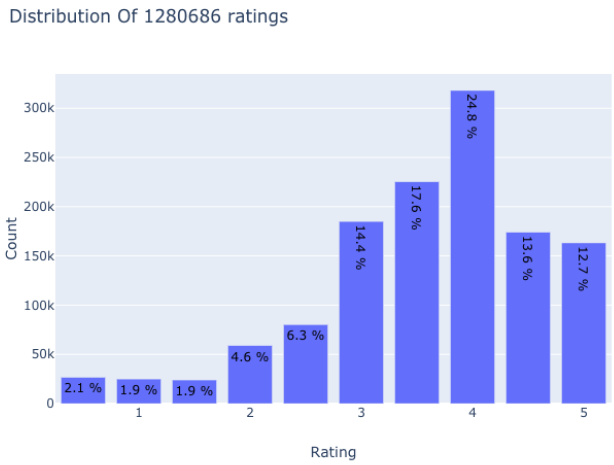

Figure 1: Distribution of ratings

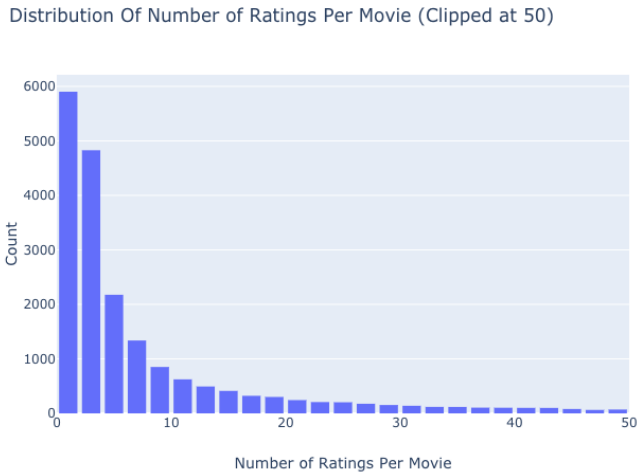

Figure 2: Distribution of ratings per movie

As shown in Figure 2, the team didn't directly use the most popular items or unpopular items to train the model. However, a popularity recommender which simply recommends the most popular k movies to users can be helpful when there is a new user.

	movieId	rating
286	318	5452
2136	2571	5221
13047	79132	5156
10773	58559	5116
2471	2959	4760
4228	4993	4252
6034	7153	4197
267	296	4077
713	858	3921
477	527	3863

Figure 3: Most popular movies

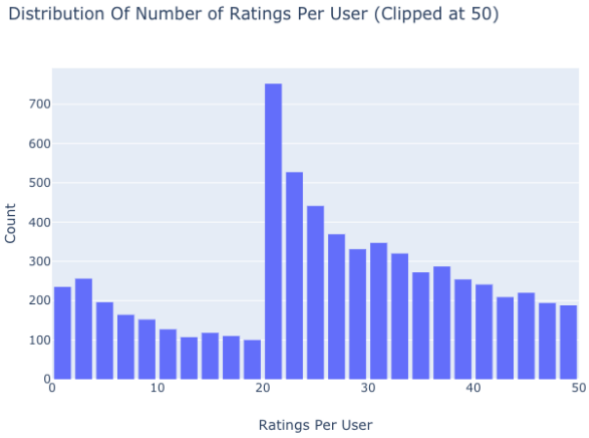

Figure 4: Distribution of number of ratings per user

The team cared about how many how many items a user has rated since inactive users will influence the accuracy of the model. However, from Figure 4, most of the users have rated more than 20 movies. In this case, those inactive users would not affect the accuracy of the model significantly.

3.2. Divide data into smaller samples

After obtaining the subset data from the raw dataset, the team then divided the subset data into smaller samples based on the number of users. Starting from 1000 users to develop the models, the team then systematically increased the sample size to the whole subset with 4 equal increments on number of users.

4. Method

The team first developed a baseline model to put a floor on the performance of the recommendation system. Then two collaborative filtering models: item-based collaborative filtering and matrix factorization were developed to improve the system.

4.1. Baseline model

The team developed a baseline model as follows:

$$\hat{r_{ij}} = \mu + b_i^{user} + b_j^{item}$$

And the bias parameter for user and item are calculated through the following objective function:

$$\text{Mininize } J = \frac{1}{2}\sum_{(i,j)\in S}(r_{ij} - \hat{r_{ij}})^2 + \frac{\lambda}{2}\left(\sum_{u=1}^{m}(b_u^{user})^2 + \sum_{j=1}^{n}(b_j^{item})^2\right)$$
$$\text{where } S = \{(i,j): r_{ij} \text{ is observed}\}$$

If user i is unknown, then the bias b_u^{user} is assumed to be zero. The same applies for item j with b_j^{item}.

The team used Stochastic Gradient Descent (SGD) to find the optimal bias parameters with a default regularization term (λ) of 0.02, learning rate of 0.005 and number of epochs of 20 by doing 3 fold cross validation over the dataset. The RMSE, MAE and run time of the baseline model were then calculated over dataset with different sizes.

As shown in Figure 5 and Figure 6, the RMSE and MAE of the baseline model tends to decrease as more users are selected to fit the model. When the number of users reach 10,000, the RMSE and MAE converge.

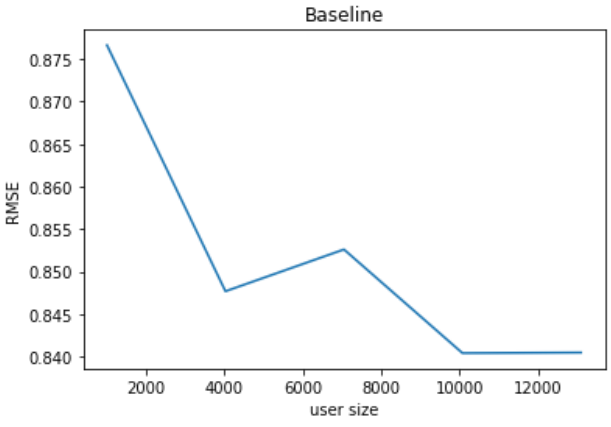

Figure 5: RMSE vs user size (Baseline)

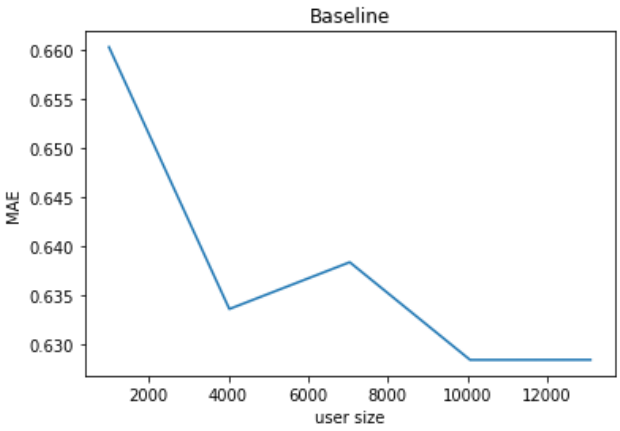

Figure 6: MAE vs user size (Baseline)

4.2. Item Based Collaborative Filtering

After comparing performance of different KNN collaborative filtering models, the team chose KNNBaseline model. KNNBaseline was improved based on KNNWithmean model, taking a baseline rating into account. The prediction is set as:

$$\widehat{r_{ui}} = b_{ui} + \frac{\sum_{j \in N_u^k(i)} sim(i,j) \cdot (r_{uj} - b_{uj})}{\sum_{j \in N_u^k(i)} sim(i,j)}$$

Looking at the dataset, the number of users is much larger than the number of movies. And the rating matrix is sparse, the team hence chose item-based collaborative filtering instead of user-based collaborative filtering for a better fit. After tuning parameters, the team further found that with K of 40 and using Pearson Similarity, the model outperformed. Then the team explored the how accuracy metrics and runt time change over sample size.

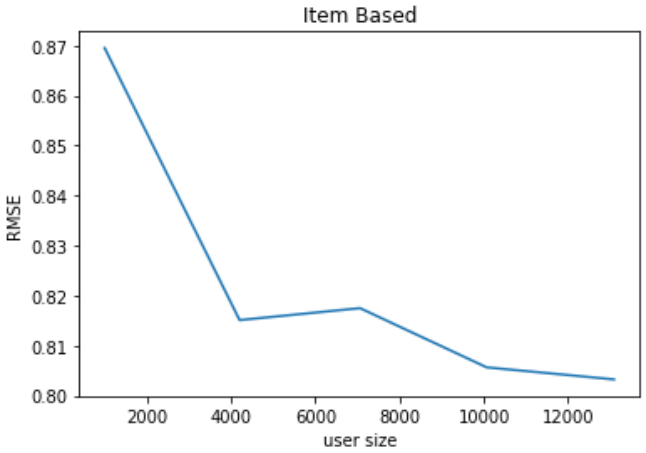

Figure 7: RMSE vs user size (Item-based)

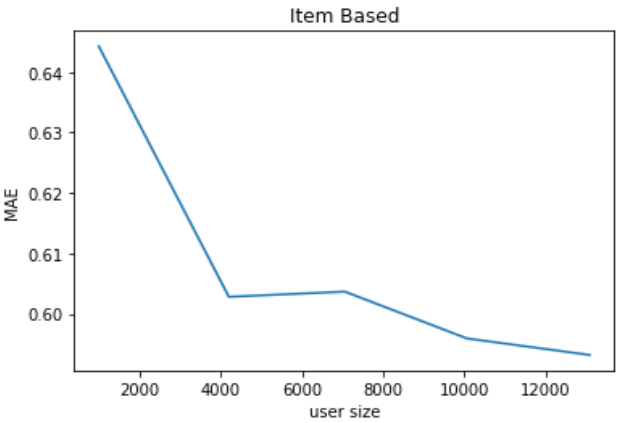

Figure 8: MAE vs user size (Item-based)

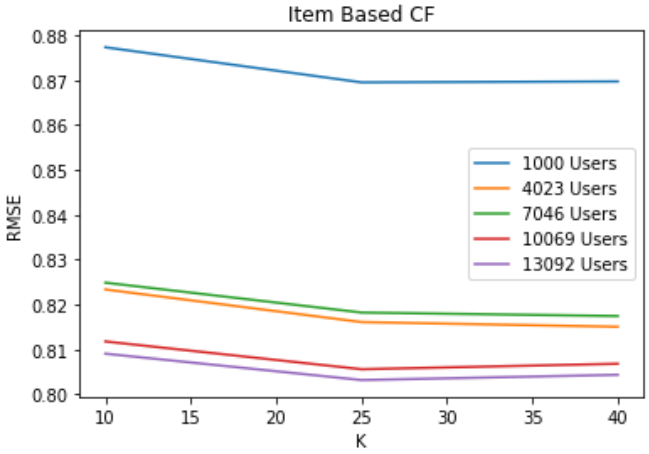

Figure 9: RMSE vs user size for different K (Item-based)

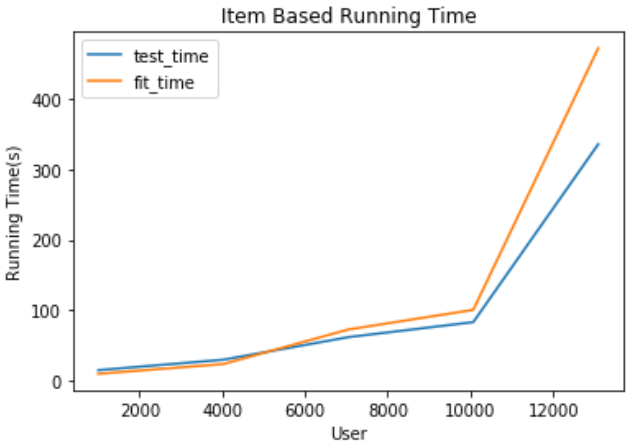

Figure 10: Run time vs user size (Item-based)

Figure 7 and Figure 8 indicate that the RMSE and MAE of the Item based collaborative filtering decrease as the dataset size becomes larger. Figure 9 indicates how RMSE changes over different value of K (number of users in the peer set) for different dataset sizes. From Figure 10, the team can conclude that the run time for this model increase when the dataset size increases. This is because more similarity between movies need to be calculated which increase the run time for the model.

4.3. Model Based Collaborative Filtering

the Model based collaborative filtering is trying to transform the ratings into a matrix that sets users as index and movies as columns. By solving the matrix factorization problem, the model has the power of approximating the ratings of any movies for a given user.

The idea behind matrix factorization is to represent users and items in a lower dimensional latent space to generalize the machine learning problem. There are a number of algorithms to do matrix factorization, for example, SVD (Singular Value Decomposition), ALS (Alternating Least Squares) etc.

For this project, the team is using ALS to solve the matrix factorization. The idea behind this algorithm is it holds user matrix (U) constant when it tries to optimize item matrix (V) and it holds the item matrix (V) constant when it is optimizing the user matrix (U). By using this iterative coordinate-descent optimization technique, the sparsity and the scalability of the large dataset could be more easily dealt with.

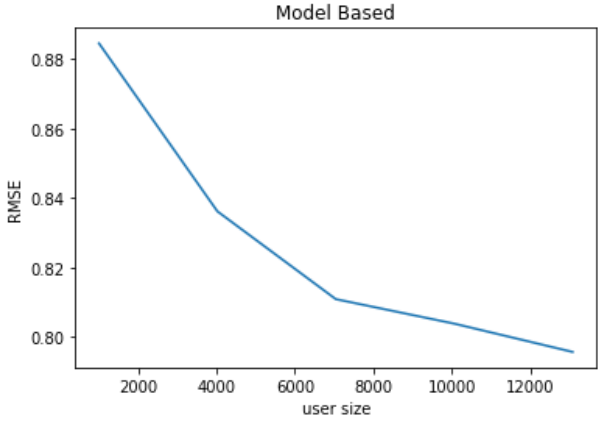

Figure 11: RMSE vs user size (Model-based)

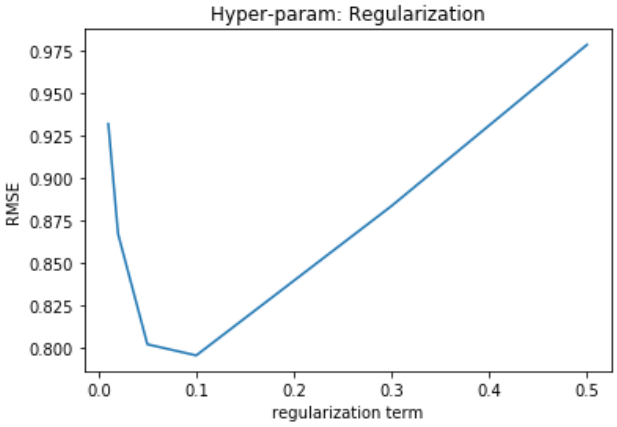

Figure 12: RMSE vs regularization term (Model-based)

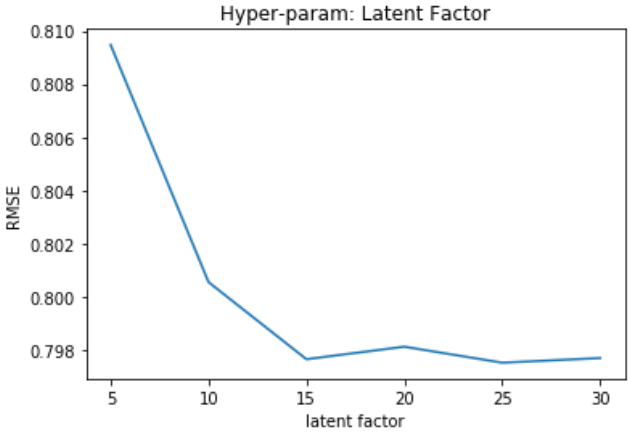

Figure 13: RMSE vs latent factore (Model-based)

First of all, the team wants to see how the RMSE would be affected with respect to different sample sizes. In order to see how the RMSE changes, the team first created a list of samples with different number of randomly picked users. Then the team fed each sample data into the model and used cross

validation to tune for the best hyper-parameters. This process was repeated iteratively for each sample with a specific number of users, and generated the graph above. As shown in Figure 11, as the sample size grows, the predictions become more accurate. In order to further investigate how the accuracy might change with respect to different hyper-parameters, the team fixed the sample size and tested a number of parameters to observe the change in accuracy of the model.

For example, in order to see how the regularization term might affect the accuracy, the team kept the sample size and well as the latent factor constant, and ran the ALS on top of a number of regularization values. Figure 12 shows that RMSE reaches minimum when it is equal to 0.1. After that, RMSE increases steadily. Figure 13 indicates how the team tried to find the best latent factor. As latent factor becomes bigger, the model seems to make better predictions. However, we do need to beware of the overfitting issue caused by a big latent factor.

4.4. Comparison

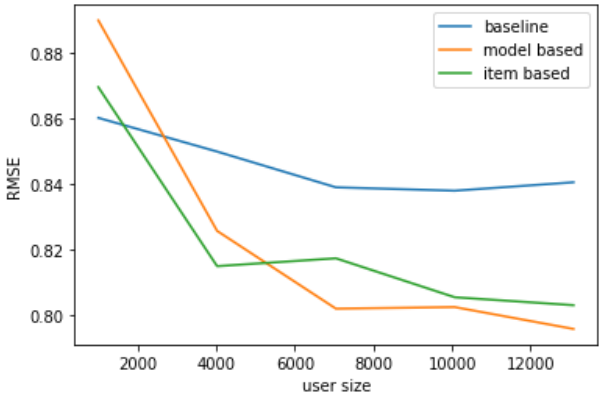

Figure 14: RMSE vs user size (All models)

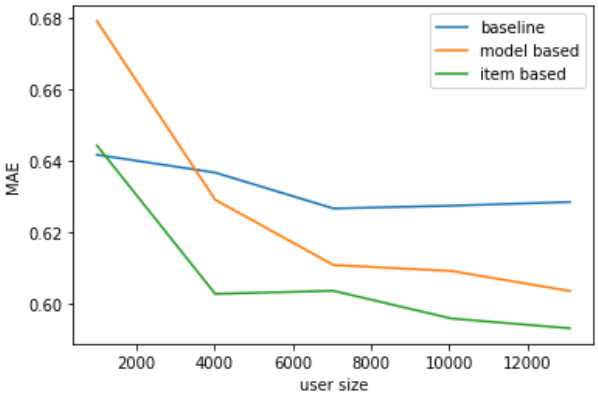

Figure 15:MAE vs user size (All models)

Figure 14 and Figure 15 show how RMSE and MAE changes for three models over different dataset sizes. When more than 4000 users were used to train the model, both item-based collaborative filtering and vanilla matrix factorization outperformed the baseline model. Moreover, there is an over decreasing trend for both metrics when the dataset size increases.

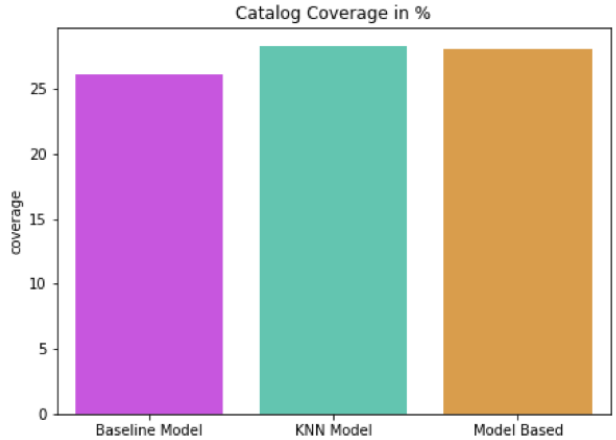

Figure 16: Catalog Coverage for all models

Figure 16 shows the coverage of three models. This is not surprising to see that baseline model has the lowest coverage (about 26%) as it is pretty naïve. And item-based collaborative filtering and vanilla matrix factorization both have higher coverage values than the baseline.

5. Application

After obtaining the models, the team randomly fit one user to them to see what would be the results and Figure 17 and Figure 18 shows the top 10 recommended movies for user with userID of 77641.

	Movie Title
Top 1	Spirited Away (Sen to Chihiro no kamikakushi) ...
Top 2	Before Sunset (2004)
Top 3	Amadeus (1984)
Top 4	12 Years a Slave (2013)
Top 5	Exit Through the Gift Shop (2010)
Top 6	Modern Times (1936)
Top 7	Citizen Kane (1941)
Top 8	Intouchables (2011)
Top 9	Twelve Monkeys (a.k.a. 12 Monkeys) (1995)
Top 10	Taxi Driver (1976)

Figure 17: Top 10 recommendations for user 77641 (Item-based)

	Movie Title
Top 1	Earthlings (2006)
Top 2	Shawshank Redemption, The (1994)
Top 3	Fight Club (1999)
Top 4	Whisper of the Heart (Mimi wo sumaseba) (1995)
Top 5	Modern Times (1936)
Top 6	Persepolis (2007)
Top 7	Amadeus (1984)
Top 8	Laputa: Castle in the Sky (Tenkû no shiro Rapy...
Top 9	Millennium Actress (Sennen joyû) (2001)
Top 10	Prestige, The (2006)

Figure 18: Top 10 recommendations for user 77641 (Model-based)

6. Discussion

In addition to vanilla Matrix Factorization, side information could be added to the model and perform collective Matrix Factorization. There are data of the relevance between movies and tags and the genres of each movie available to use. By adding these side information, the model can be more accurate and useful.

The RMSE and MAE of the models are pretty low for both of the models for the recommendation system to provide accurate results and the coverage values are high enough to provide various movies to user. Therefore, the team is confident to say that the recommendation system meets the objectives for the project. However, there are some watch outs need to be considered before it can be put into production at a real company.

The first one is the user cold start. New users to the website may not register upon their first arrivals. Thus, there is no historical information for the system to provide recommendations for those users. One potential solution is to identify new users by using cookies. Once a user is identified as a new user, the system can recommend the most popular 10 movies to them. Another one is item cold start. When there is a new movie, it is less likely to be recommended to users according to item-based collaborative filtering. In this case, the company needs to collect information about this movie such as genres and tag, and build a content-based collaborative filtering to provide recommendations to users.