COMP9417 Project Report

Topic 3.4: Recommender system using collaborative filtering

z3492782 Li YU

z5106859 Yiming WANG

Contents

Introduction **3**

Method **4**

The MovieLens Dataset 4
Data Exploration 4
Type of Recommendations Engines 4
Similarity Matrix 5
Predictive Methods 6
New User Prediction 7
Near-Rating-Level Round-Off 7
Error Metrics 8

Experimentation Process **9**

Train and Test data splitting 9
Similarity Matrix Computing 9
Rating Prediction 9

Results **12**

The best Model with parameters 12
Recommendation 12
Running time 13

Conclusion **13**

References **13**

Introduction

High-quality recommendation systems are the main drivers for success of many online services such as Netflix, Amazon and eBay among others. A predictive recommender has a direct impact on a growing number of businesses.

The project implemented a predictive Movie Rating Recommendation with Collaborative Filtering. The 5-star rating parameter in MovieLens dataset is a critical parameter in determining whether a user likes a movie or not.

Collaborative filtering movies recommendations based on the knowledge of users attitude to items which using the wisdom of the crowd to recommend items.

The task of this project is to predict all ratings for testing data set based in training data set. Finding top k (e.g. k = 5,10,15,... 70, 75, 80, 100…) similar users/movies is the key to establish the best model. The goal is to minimize the MAE (mean average error) and RMSE (root mean square error) when predicting the ratings on a test set. Then we can choose top items as the recommendations for users.

This report will take on the following structures:

- Exploration of the data set used for the project.
- Explanation of method using for predict user rating for movies.
- Brief discussion on the performance metric used in model evaluation.
- Description and evaluation of the k-neighborhood with baseline predictor method.
- Conclusion and future prospects.

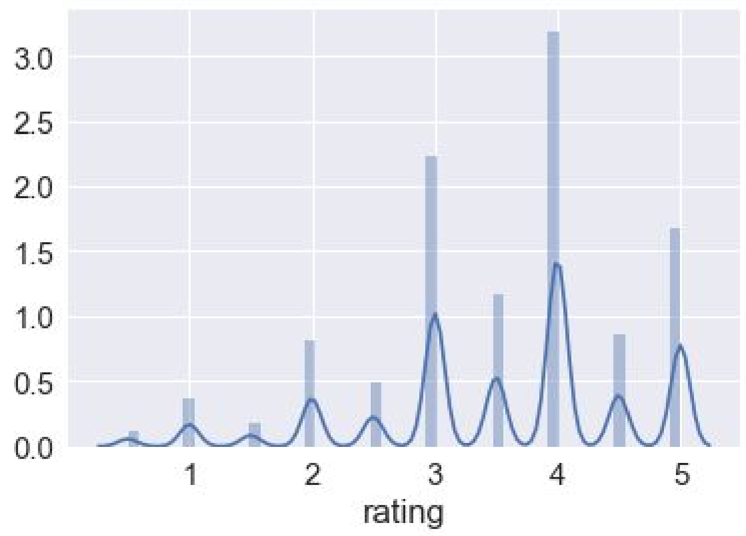
Method

1. The MovieLens Dataset

MovieLens is one of the most common datasets available on the internet for building a Recommender System. Due to the limited computation power of PC, the main version of the dataset that the project is working with (ml-latest-small) contains 100,000 ratings and 1,300 tag applications applied to 9,000 movies by 700 users. Last updated 10/2016. All users selected had rated at least 20 movies. Each user or movie is represented by an id. The original data are mainly contained in three files, movies.csv, ratings.csv and tags.csv.

2. Data Exploration

The figure below shows the distribution of movie ratings. Most users are quite generous when rating movies. The mean rating overall data set is 3.5399 on a scale of 5. There is one problem that 5-star rating isn't a wonderful indicator as users may have different rating styles. For instance, user A always rating 5 for his favorites, whereas user B only gives 3.



3. Type of Recommendations Engines

Memory-Based Recommendation Model(Collaborative Filtering)

Based on the choice of reference characteristics, a recommendation system could be based on content-based approach or collaborative filtering approach. In this project, we build a recommendation system based on Multiple Collaborative Filtering. There are two main types of Memory-based Collaborative Filtering algorithms:

1) User-User Collaborative Filtering:

This method starts by finding alike users based on similarity and recommend movies which user's look-alike has chosen in past. This algorithm requires to compute every user pair information which takes time. So, for large dataset in big platforms, this algorithm is hard to implement without a very strong parallelizable system.

2) Item-Item Collaborative Filtering:

Similar to User-User Collaborative Filtering method, this method computes the similarity matrix of all the movies. It then estimate the missing ratings based on top-k similar items to the target one. There are three major problems about this method. The accuracy highly depends on the density of the rating matrix. When the size of the data set grows, the similarity matrix will grow faster which is computationally expensive. And when it comes to live system, the items' ratings will change frequently which requires the similarity matrix to be updated frequently as well. Thus, it's relatively hard to maintain the similarity matrix.

Content-Based Recommendation Model

Content-Based filtering method is implemented with a short description of the movie or some keywords and tags written down by the users. It will also build a profile for each user so that the algorithm could recommend items similar to the user's profile. Due to the short duration of this project, we didn't implement this method. It could be a future implementation direction for this topic.

4. Similarity Matrix

Building similarity matrix is a must for collaborative filtering. The user-similarity matrix will consist of the distance metrics that measures the similarity between any two pairs of users. Likewise, the item-similarity matrix will measures the similarity between any two pairs of movies. There are many possible ways of measuring this matrix. However, Jaccard metric would not be the most effective, as it would not take into account the 0 to 5 ratings assigned to each movie. In this case the Pearson Coefficient and Cosine similarity measure are the most reliable metric.

1) User-User k-neighborhood model (combined with baseline predictor) :

In this project, to measures the Similarity between two user vectors, **Pearson Coefficient** would be the most reliable metric for User-User Collaborative Filtering to remove the presence of any possible user bias when predicting (i.e. a user has the tendency to rate everything higher than it's "true" rating).

Pearson Coefficient:

$$sim(x,y) = \frac{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})(r_{ys} - \overline{r_y})}{\sqrt{\sum_{s \in S_{xy}} (r_{xs} - \overline{r_x})^2} \sqrt{\sum_{s \in S_{xy}} (r_{ys} - \overline{r_y})^2}}$$

The advantage of the above-defined similarity is that the differences in the rating scale between different users are taken into consideration.

For given user x, we begin by calculating the similarity between this user and all other users. The we select its k-nearest neighbors based on the similarity measure. For a movie, it's predicted rating is not only computed based on a weighted combination of the k-nearest neighbor ratings, but also the baseline predictor will correct the negative predictions by averaging the distance from the baseline predictor.

Issues the project encounter that users with no ratings history except for the one rating the system trying to predict, or users whose neighbors had never rated the movie for that we want to predict. The model also suffers from scalability problems, as the user base grows significantly in the larger dataset (ml-latest 26,000,000 ratings and 750,000 tag applications applied to 45,000 movies by 270,000 users. Includes tag genome data with 12 million relevance scores across 1,100 tags. Last updated 8/2017).

2) Item-Item Collaborative Filtering:

We tried to apply the same measurement above to Item-Item Collaborative Filtering method but only realized that it won't work. The rating matrix is very sparse. It's possible that some of the movies in the data set only have a limited amount of rating records. When calculating the average rating of a certain movie, it's likely that all the ratings are of the same value, given that it will only have a few ratings after all. As a result, the similarity will come to 0 which mathematically means the two movies are not related. However, this is not true. And in this case, some movies will have no similar movies at all no matter positively or negatively. To solve this problem, we decide to apply cosine similarity which is basically a simplified version of the pearson methods. Instead of subtract the mean value, we calculate the cosine value of the two rating vection directly as below.

Cosine Coefficient:

$$sim(x,y) = cos(r_x, r_y) = \frac{r_x \bullet r_y}{||r_x|| \cdot ||r_y||}$$

5. Predictive Methods

Given the user-movie ratings matrix, we aim to fill in the missing ratings in our sparse matrix. In practice we get better estimates if we model deviations:

Global Baseline Estimate:
In case that the target user or movie doesn't have any similar users or movies, we introduce a global baseline estimate. The global baseline estimate is made up of overall mean movie rating, rating deviation of user x and rating deviation of movie i.

Local neighborhood (CF/NN):

We applied both User-User Collaborative Filtering and Item-Item Collaborative Filtering, then we combined them together. After calculating the similarity matrix, we find top-k nearest neighbours for each user/movie.

Final Estimate:
Looking at nearest neighbor and including the baseline predictor to correct the negative predictions by averaging the distance from the **baseline predictor** is helpful to boost the system predictive performance. We begin by implementing the following model to the dataset:

$$r_{xi} = b_{xi} + \frac{\sum_{j \in N(i;x)} s_{ij} * (r_{xj} - b_{xj})}{\sum_{j \in N(i;x)} s_{ij}}$$

$$\textit{baseline estimate for} \quad b_{xi} = \mu + b_x + b_i \qquad \mu = \textit{overall mean movie rating}$$

$$b_x = \textit{rating deviation of user } x$$

$$b_i = \textit{rating deviation of movie } i$$

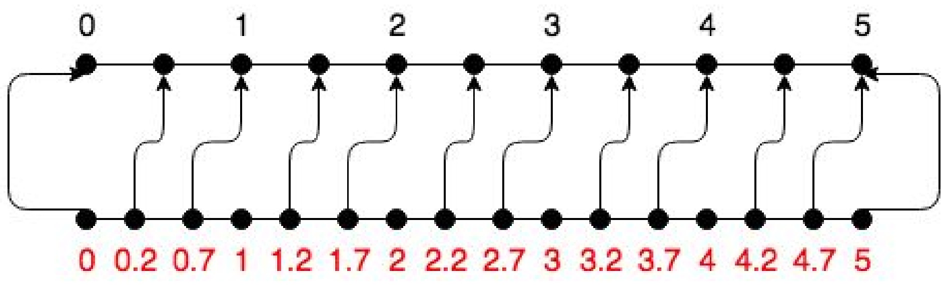
In this model, in order to remove the presence of any possible bias when predicting (i.e. a user has the tendency to rate everything higher than it's "true" rating), the mean rating given by the user and the movie has been taking into account.

6. New User Prediction

When there exist a newcomer (a user without any existent ratings), it is very difficult to predict his rating on any movie. In order to minimize the RMSE in this case, we use the movie mean as his predicted rating. The prediction of new user i's rating on Movie j is the average rating of existent ratings on Movie j given by all other users.

7. Near-Rating-Level Round-Off

In this project, each predicted rating will be round off if it close enough to an level of rating. During the implementation, we round off the prediction if its distance to the nearest level of rating less than or equal to 0.3. It has been observed that this approach can improve the performance of the recommendation system.



8. Error Metrics

The error metric we look at throughout this project is probably the most popular one when evaluating recommender systems. We train the model on 80% of the data and then test its accuracy on the remaining 20%. The score is mainly measured with mean root squared error (RMSE) and mean Absolute Error (MAE).

RMSE: The Root Mean Squared error is a prediction accuracy metric that answer the question of how close the prediction ratings are to the true ratings. It uses squared deviations and so larger errors tend to get amplified.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (\widehat{r_i} - r_i)^2}{n}}$$

MSE: The Mean Squared error is a prediction measures the average magnitude of the errors in a set of predictions. When the RMSE is less than 1 (i.e. our best model achieve RMSE 0.9393, MSE 0.8825), MSE is ideal to illustrate an error value without taking Root operation.

$$MSE = \frac{\sum_{i=1}^{n} (\widehat{r_i} - r_i)^2}{n}$$

MAE: The mean absolute error is a prediction measures average magnitude of the errors in a set of predictions, without considering their direction.

$$MAE = \frac{\sum_{i=1}^{n} |\widehat{r_i} - r_i|}{n}$$

Experimentation Process

1. Train and Test data splitting

We train the model on 80% of the data and then test its accuracy on the remaining 20%. 5-folder Cross Validation with in the experimentation process.

2. Similarity Matrix Computing

For the userCF, **Pearson Coefficient** would be the most reliable metric for User-User Collaborative Filtering to remove the presence of any possible user bias when predicting (i.e. a user has the tendency to rate everything higher than it’s “true” rating).

For the itemCF, **Cosine Similarity** will be a better choice to avoid some cases where a movie only has a few ratings and all these ratings are of the same value. It might be considered not similar to any other ratings in the matrix with Pearson Coefficient applied.

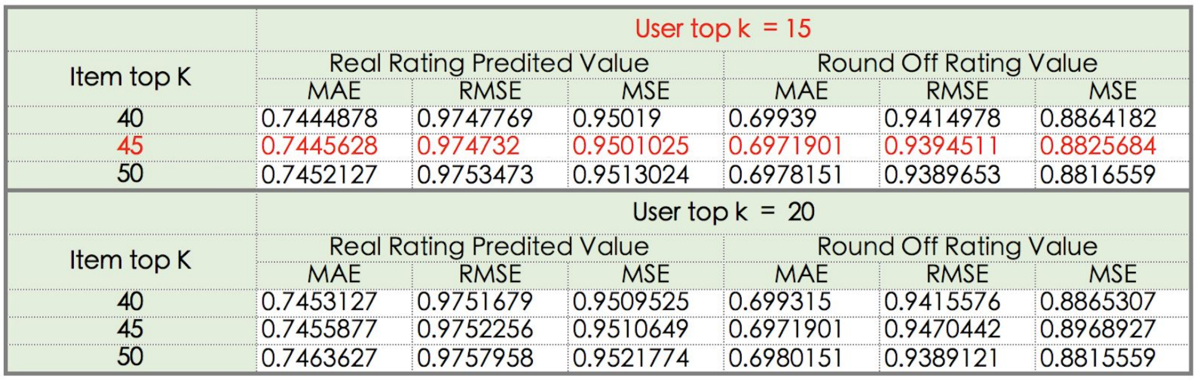
For both userCF and itemCF algorithm, we use K-fold cross-validation to choose the best parameters for the model. In particular:

- choose top K users in the user CF implementation from [5,10,15,20,25,30,35,40,35,50,55,60,65,70,75,80,90,100,120,140,160]
- choose top K items in the item CF implementation from [5,10,15,20,25,30,35,40,35,50,55,60,65,70,75,80,90,100,120,140,160]

3. Rating Prediction

As different K neighborhood size will influence the experiment outcome with the same data set. We use 5-folder Cross Validation to get the process and data comparison list.

The best model during experimentations shows up with 15 (k value) nearest users and 45 (k value) nearest movies shows the best MAE 0.697, RMSE 0.939 and MSE 0.882.

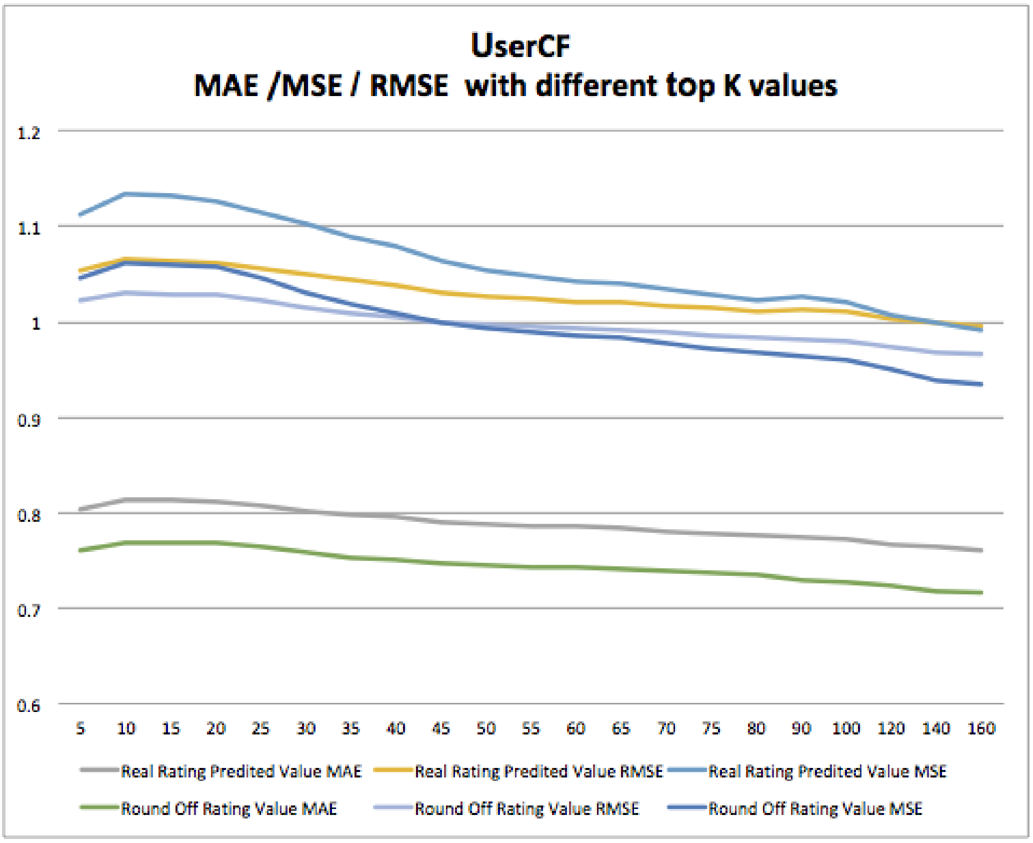


	User top k = 15					
Item top K	Real Rating Predited Value			Round Off Rating Value		
	MAE	RMSE	MSE	MAE	RMSE	MSE
40	0.7444878	0.9747769	0.95019	0.69939	0.9414978	0.8864182
45	0.7445628	0.974732	0.9501025	0.6971901	0.9394511	0.8825684
50	0.7452127	0.9753473	0.9513024	0.6978151	0.9389653	0.8816559

	User top k = 20					
Item top K	Real Rating Predited Value			Round Off Rating Value		
	MAE	RMSE	MSE	MAE	RMSE	MSE
40	0.7453127	0.9751679	0.9509525	0.699315	0.9415576	0.8865307
45	0.7455877	0.9752256	0.9510649	0.6971901	0.9470442	0.8968927
50	0.7463627	0.9757958	0.9521774	0.6980151	0.9389121	0.8815559

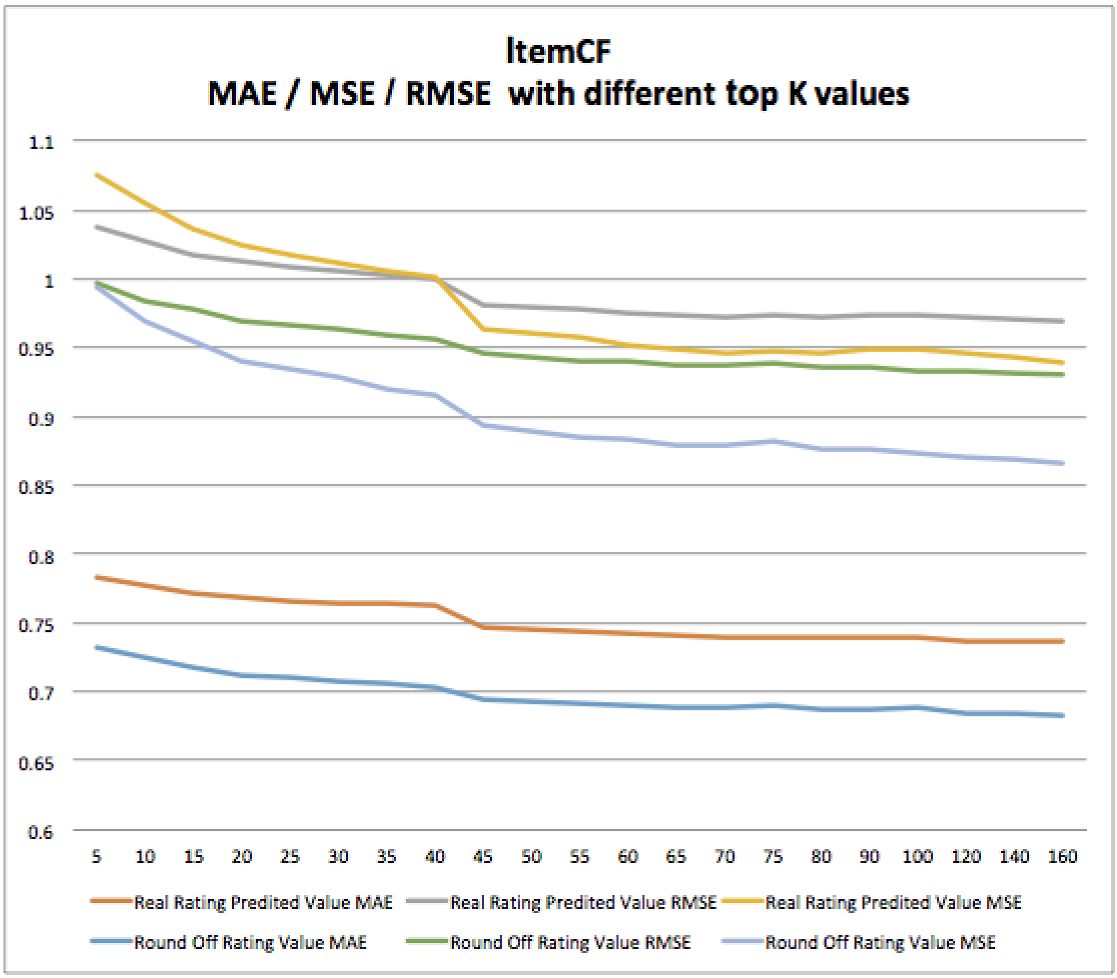
As shown in the figures below, with the top K values increasing, the accuracy of recommendation system has been improved because of our prediction method with baseline predictor. With more users/movies, the predicted rating has been benefited by not only it similar neighbours, but also some unlike users/movies. As negative similarity values can adjust this predicted rating more accuracy.

User top K	Real Rating Predited Value			Round Off Rating Value		
	MAE	RMSE	MSE	MAE	RMSE	MSE
5	0.8037348	1.0546833	1.1123569	0.759812	1.0223002	1.0450977
10	0.8133093	1.0650673	1.1343683	0.7688616	1.0305445	1.0620219
15	0.8126844	1.0637579	1.1315809	0.7688116	1.0290637	1.0589721
20	0.8109595	1.0613758	1.1265187	0.7674866	1.0282192	1.0572346
25	0.8069347	1.0553468	1.1137568	0.7641118	1.0222513	1.0449978
30	0.8017349	1.0500511	1.1026074	0.7580871	1.0152394	1.030711
35	0.7981851	1.0434237	1.0887331	0.7534373	1.0090399	1.0181616
40	0.7949853	1.038633	1.0787586	0.7501875	1.0048318	1.009687
45	0.7904855	1.030975	1.0629094	0.7478126	0.9992247	0.9984501
50	0.7869857	1.0264184	1.0535348	0.7449878	0.9966132	0.9932378
55	0.7857357	1.0237481	1.0480601	0.7433378	0.9945668	0.989163
60	0.7851857	1.0210952	1.0426354	0.7425129	0.9926483	0.9853507
65	0.7837858	1.0202992	1.0410104	0.7410879	0.9915207	0.9831133
70	0.780636	1.0165559	1.0333858	0.7388631	0.9885854	0.9773011
75	0.7788611	1.0143957	1.0289986	0.7371881	0.9858569	0.9719139
80	0.7765112	1.01157	1.0232738	0.7353632	0.9836547	0.9675766
90	0.7743863	1.012811	1.0257862	0.7293135	0.9816832	0.9637018
100	0.7719114	1.010049	1.020199	0.7277136	0.9799627	0.960327
120	0.7667117	1.0038424	1.0076996	0.7231388	0.9744434	0.94954
140	0.7636868	0.9999688	0.9999375	0.7183891	0.9685895	0.9381656
160	0.759937	0.9955214	0.9910629	0.7162642	0.9662186	0.9335783



Anther point has been found is that item-based collaborative filtering shows a better performance than user-based collaborative filtering. This is because the data set has larger number of movies.

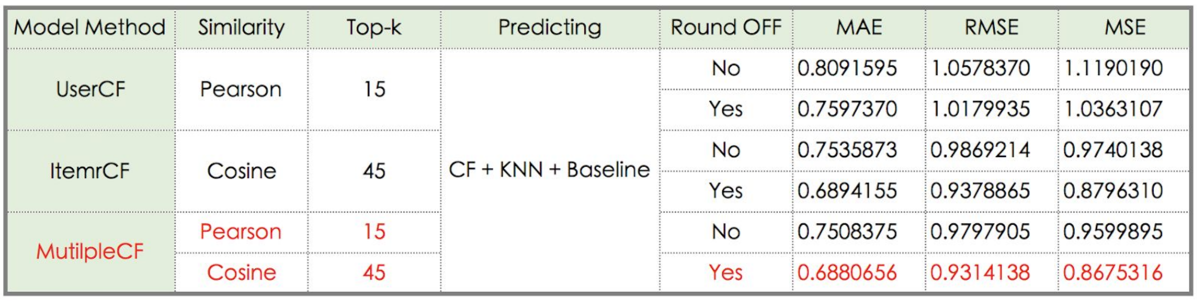
Item top K	Real Rating Predited Value			Round Off Rating Value		
	MAE	RMSE	MSE	MAE	RMSE	MSE
5	0.7830358	1.036995	1.0753587	0.7320634	0.9971461	0.9943003
10	0.7766862	1.0275138	1.0557847	0.7247138	0.9846073	0.9694515
15	0.7711114	1.0176926	1.0356982	0.7173141	0.9773956	0.9553022
20	0.7678366	1.0124901	1.0251362	0.7113894	0.969299	0.9395405
25	0.7660867	1.0086559	1.0173866	0.7106895	0.9666518	0.9344158
30	0.7645368	1.005466	1.010962	0.7080896	0.9634786	0.9282911
35	0.7644618	1.0027336	1.0054747	0.7054647	0.9590902	0.919854
40	0.7620869	1.0004811	1.0009625	0.7033398	0.9567349	0.9153417
45	0.7461127	0.9812502	0.9628519	0.6945403	0.9456045	0.8941678
50	0.7453377	0.9799436	0.9602895	0.6925404	0.942745	0.8887681
55	0.7441128	0.9782521	0.9569772	0.6908655	0.9409334	0.8853557
60	0.7421629	0.9756164	0.9518274	0.6900155	0.9399366	0.8834808
65	0.740913	0.9743729	0.9494025	0.6879156	0.9374066	0.8787311
70	0.739163	0.9727938	0.9463277	0.6883656	0.9376066	0.879106
75	0.739913	0.9734873	0.9476776	0.6892405	0.9388122	0.8813684
80	0.739513	0.9723954	0.9455527	0.6867407	0.9358652	0.8758437
90	0.739563	0.9742446	0.9491525	0.6864157	0.9357383	0.8756062
100	0.7387381	0.973789	0.9482651	0.6878886	0.9336768	0.8736893
120	0.7371131	0.9724468	0.9456527	0.6841658	0.9327146	0.8699565
140	0.7367132	0.9708645	0.9425779	0.6840658	0.9320711	0.8687566
160	0.7365632	0.9691249	0.939203	0.6821159	0.9305814	0.8659817



Results

1. The best Model with parameters

It is obviously that with **baseline predictor** method to predicting ratings, the accuracy of recommendation system will been improved by increasing the value of k-nearest neighborhood. The MAE, MSE, RMSE of all three collective filtering which are userCF, itemCF, multipleCF has been decreased dramatically.

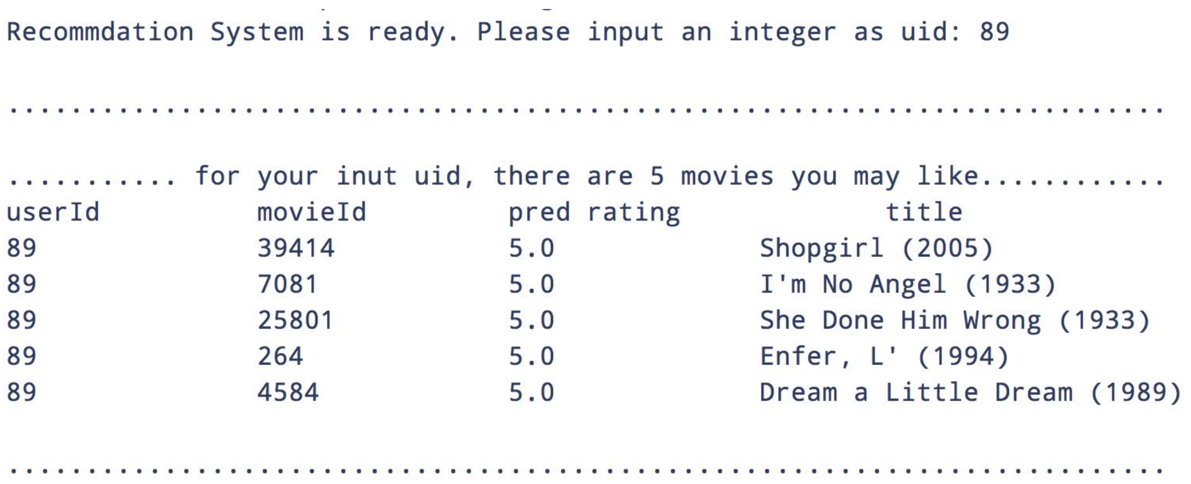


Model Method	Similarity	Top-k	Predicting	Round OFF	MAE	RMSE	MSE
UserCF	Pearson	15	CF + KNN + Baseline	No	0.8091595	1.0578370	1.1190190
				Yes	0.7597370	1.0179935	1.0363107
ItemrCF	Cosine	45		No	0.7535873	0.9869214	0.9740138
				Yes	0.6894155	0.9378865	0.8796310
MutilpleCF	Pearson	15		No	0.7508375	0.9797905	0.9599895
	Cosine	45		Yes	0.6880656	0.9314138	0.8675316

At all circumstance, item-based algorithms provides better quality than user-based algorithms.

2. Recommendation

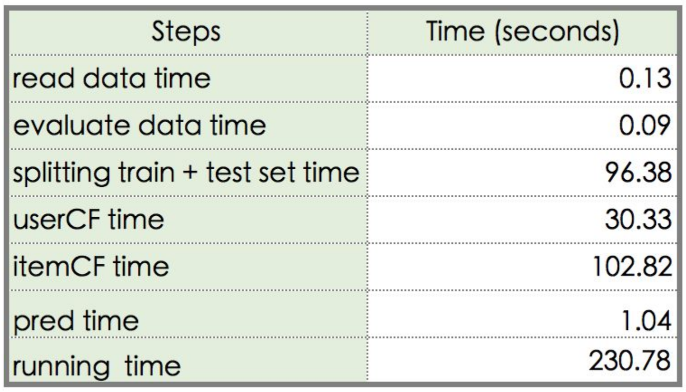
Based on the model that we've built and the optimised parameters, we now can recommend movies for a given user.



```
Recommdation System is ready. Please input an integer as uid: 89

.....................................................................

........... for your inut uid, there are 5 movies you may like............
userId          movieId         pred rating             title
89              39414           5.0             Shopgirl (2005)
89              7081            5.0             I'm No Angel (1933)
89              25801           5.0             She Done Him Wrong (1933)
89              264             5.0             Enfer, L' (1994)
89              4584            5.0             Dream a Little Dream (1989)

.....................................................................
```

3. Running time

The whole program package has 3 files. Mainly run recommender.py file and it will call userCF.py and itemCF.py. The entire process of making recommendation matrix is around 230 seconds.



Steps	Time (seconds)
read data time	0.13
evaluate data time	0.09
splitting train + test set time	96.38
userCF time	30.33
itemCF time	102.82
pred time	1.04
running time	230.78

Conclusion

In this project, we present two collaborative filtering algorithms with baseline estimator for recommendation system and test the performance of each algorithm. At all circumstance, item-based algorithms provides better quality than user-based algorithms. Combined both user-based and item-based collaborative filters gives the best accuracy of the recommendation system. Finally, a MAE of 0.688 is achieved.

The main issues faced by Collaborative Filtering in this project were the sparseness of data and the cold-start problem (movie cannot be recommended unless it hasn't been rated before). Given the amount of data of MovieLens, a possible solution would be to combine collaborative recommendation systems with content-based filters to exploit the information of the items already rated.

References

1. Mining of Massive Datasets - Stanford lectures
2. Programming Collective Intelligence - by Toby Segaran
3. The BellKor Solution to the Netflix Grand Prize - by Yehuda Koren