[[1]](#footnote-1)

Movie Recommender System: Common and Hybrid Algorithms Review and Implementation(April 2019)

Yongjian Zeng(20773103), Haozheng Zhao(20781861), and Zhehao Xu(20762577)

***Abstract—As more and more movies are produced in recent decades, it becomes hard to find movies that matches individuals’ favors as more time is going to spend on filtering items from the large movie dataset. Therefore, to solve this problem for users, a recommendation feature is proposed which generates a list of movies from the current database that the user will prefer to watch most likely. The recommendation is computed and generated automatically based on machine learning. In this project, several popular recommending algorithms are discussed and implemented to compare which one gives the best performance in terms of accuracy and coverage.***

*Index Terms—Collaborative filtering, Genre correlation, Hybrid model, Latent factor model, Movie recommendation.*

This work was supported in part by the University of Waterloo”

Yongjian Zeng is with Electrical and computer engineering department, University of Waterloo, Canada (e-mail: y83zeng@uwaterloo,ca).

Haozheng Zhao is with Electrical and computer engineering department, University of Waterloo, Canada (e-mail: y83zeng@uwaterloo,ca).

Zhehao Xu is with Electrical and computer engineering department, University of Waterloo, Canada (e-mail: y83zeng@uwaterloo,ca).

# Introduction

M

ovie recommender system is always an important feature for movie-collected websites because users will soon become confusing and frustrating when the amount of options they can choose from increases gradually. The idea of the system is to recommend a list of movies from the database that they will have a large possibility to watch using machine learning techniques. There are several popular algorithms currently used commercially and academically. A good movie recommender system feature can attract more users to continue to watch movies on the website as it gives them a better experience instead of spending lots of time on finding movies they like in a large dataset pool. In this project, 4 popular algorithms are selected to study which are collaborative filtering, latent factor model, genre correlation, and top list N. The goal is to implement each and compare results to see which one matches users’ favour the most as the best solution to recommend movies to users.

# Literature Review

Collaborative filtering is widely used techniques in recommendation. It considers only explicit ratings of users and make predictions[5] . Furthermore, there are two types of collaborative filtering techniques frequently used in the recommendation system domain such as model-based and memory-based collaborative filtering. Model-based method develops a user model utilizing ratings of each user to evaluate the expected value of unrated items. On the other hand, memory-based method utilizes similarity measure computed from the explicit user rating to identify neighborhoods and perform prediction. Item-based collaborative filtering and Latent factor is example of these two kinds of methods.

However, collaborative filtering predicts random rating values because of the stochastic prediction methodology, dynamic rating data, and subjectivity users. Thus, an accurate rating prediction is an insignificant methodology in the recommender system domain. Still, researchers are constantly attempted to improve the performance of collaborative filtering in terms of accuracy metrics. In this regard, Wang and Yu applied genetic algorithm to cluster users and improve the performance of User-based collaborative filtering [6]. Rahul Katarya discussed a method which makes use of k-means clustering by adopting cuckoo search optimization algorithm, and a hybrid recommender system which utilized k-means clustering algorithm with bio-inspired artificial bee colony (ABC) optimization technique [1][2]. Sujoy Bag proposed a new similarity metrics: Relevant Jaccard similarity to improve the precision of finding neighbors. Instead of predicting movie ratings based on user's historical behavior [4]. Algorithms do recommendation based on movies genres is presented by Sang-Min Choi [3]. This algorithm tackles the cold start problem.

In the literature, various traditional similarity approaches like cosine similarity, Pearson’s correlation, Jaccard similarity etc. and several data mining techniques are presented to classify nearest neighbors and generate recommendations. Although, most of the techniques are implemented to predict movies ratings and it can’t reflect whether it can recommend adequate movies to user. The purpose of this paper is research how to implement these algorithms to do Top-N recommendation and evaluate the performance.

# Problem Definition

The problem we aim to solve is to provide a practical movie recommender system to users based on user’s history behavior and it can be categorized as a regression problem. The outcomes are predicted scores on each movie in the database for the user and movies with the highest predicted scores will be selected to the recommendation list. In order to build more intelligent recommendation system, we try to implement several algorithms and compare the result. Then we will evaluate the recommendation list with some metrics.

There are multiple problems faced during the development which are time limitation, size of training dataset, and accuracy optimization. Due to the large amount of datapoints it has, the training phase becomes time-consuming.

# Methods And Approach

In this part, we will introduce the principle and details of algorithms we used in our research.

## Item-based collaborative filtering(ICF)

ICF is memory-based collaborative filtering. It recommends items similar to user’s previous preference. To be more specific, it will calculate both the similarity between each items’ feature values and analyzing user’s behavior.

There are several steps to implement ICF. Firstly, build a user-movie rating table and calculate each items’ similarity. Intuitively, the similarity between items can be derived from an equation like what Amazon’s recommendation system says: “Customers Who Bought This Item Also Bought”. But, if an item gets too popular, there is chance that many users would like the popular item. Or, if a user is too active, the user may choose items not out of his interest. To avoid these problems, we use Jaccard equation and introduce a “Inverse User Frequency” coefficient. To further improve accuracy, a time - dependent decay term is introduced.

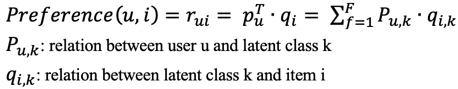
Then, system can recommend items according to the result of equation(1) [5]. For each movie in the dataset, Wji is the similarity between it and the one that is going to be predict, Rui equals to 1 if the user rated it before, otherwise Rui equals to 0. Also, only movies rated higher than 3 will be considered, so that the chance of recommending boring movies would be minimized.

 (1)

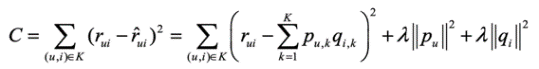
## Latent factor model (LFM)

Latent factor is a kind of model-based collaborative filtering. The basic idea of LFM is to link user and items with latent factor. Usually, editors will classify each item according to their own opinion. But editors can not represent all users’ personal opinion. Also, it is hard for editors to classify an item to multi classes or give an item a multi-dimensional classification. Accordingly, it is better to classify items based on data and automatically classify them. LFM will classify items based on users’ behavior, rather than editor’s personal opinion.

For LFM, the user’s rating of one item can be derived from equation (2). Pu,k measures the relation between user U and latent cluster K, while Qi,k measures the relation between latent cluster K and item I [5].

 (2)

To train the model, a dataset of both positive and negative samples is needed, so that researcher can know what the user like and dislike. However, a dataset usually just has positive samples. Accordingly, the first thing is to find negative samples. There are two important rules for find negative sample. In the Yahoo! Music KDD Cup 2011, researchers found that: 1. the amount of positive & negative sample should be similar; 2. negative samples should be selected from items that is popular but not rated by this user, which shows that the user is really not interested in this item. With both positive and negative samples acquired, a matrix P and Q can be generated randomly. Then, optimized P and Q can be derived from the loss function C, equation (3) [5]. Then, take the optimized and back into the preference function, the user’s potential interest of an item can be derived.

 (3)

## Genre-correlation based model

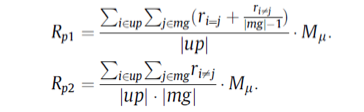
Genre-correlation based model do recommendation based on user preference and movies genres. It firstly computes genre correlation from the training set. Genre correlation is calculated based on genre combinations of each movie in a database. Each movie has a genre combination composed of at least one genre. In other words, each movie has a genre combination composed of at least one genre. In other words, each movie has a genre combination composed of at least one genre. This method firstly selects a criterion genre and counts the number of other genres for each movie. For example, if movie A has the genre combination of G1, G2 and G5, then G1 is selected as a criterion genre and we increase the combination counting with G2 and G5 by 1. Next, G2 is selected as a criterion and we increase the combination counting with only G5 by 1 again. We

repeat this procedure for all movies in the database and get the genre correlations by percentage.



Fig. 1. Count of rating with different values

Next, the genre correlations are applied with the equation (4) [3]. If the selected genre of the movie is one of the user’s preferred genres, the first equation is used. Otherwise, the second equation is used. Here, ‘up’ are preferred genres of the user, and ‘mg’ is genre combinations for each movie, ‘M’ is average rating of movie. The result of this equation is the recommendation point for the movie.

 (4)

Then, the system ranks movies according to the recommendation points and recommends top movies accordingly.

## Top list Recommendation

The idea of top list recommendation is quite straightforward . It is used to generating top movie list. Firstly, calculate the weighted rating of movies accordingly to the following function. The first term is the mean rating of the movie times its weight. While the other term is the global mean rating of all movies times its weight. Then, rank all movies according to their weighted rating and recommend top N movies accordingly.

 (5)

Here, R is the mean rating of the movie; v is the total number of ratings of the movie; m is the min number of ratings needed to be ranked as TOP 250 (500 for our experiment); C is the global mean of all ratings in the dataset (3.55 for our experiment).

# Experiments And Analysis

TABLE I

Precision/Coverage for different models with certain parameters

|  |  |  |
| --- | --- | --- |
| Model | Precision | Coverage |
| ICF-IUF-time | 26.0% | 42.7% |
| LFM | 24.4% | 20.1% |
| Genre correlation-based | 7.2% | 30% |
| Hybrid method 1 | 26.7% | 38.2% |
| Hybrid method 2 | 24.3% | 43.7% |

In this part, we will introduce the dataset we used. And conduct different experiments to evaluate the performance of algorithms above.

## Dataset

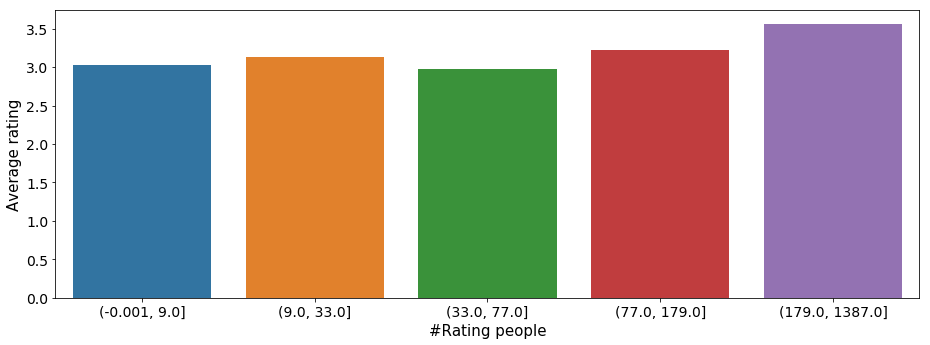


Fig. 2. Relationship between number of rating people and average rating value.

The dataset used in the project is the 1M MovieLens dataset. It contains 1 million rating records from 6000 users on 4000 movies and assigned to a discrete scale of 1–5. Besides ratings data, it contains movie file which has all the movies with their titles and genres recorded along with a unique movieid for identification. Fig. 1 shows the count of different ratings. We can see that most of rating is greater than 2, so it reflects that in most of the case, user will only watch and rate movies that they are interested in. And even we take movies will rating higher than 2 as users’ favorite movies, we will still have enough data to train our model. Fig. 2 indicates the relation ship between the number of rating people and average rating value. The pictures provide information that popular movies are also good movies. It is useful when we try to filter movies in our algorithms.

In order to speed up the experiment, we shrunk the size down to 3500 users and 650 movies so that we can implement and enhance the algorithm more fastly.

## Evaluation Criteria

In order to evaluate whether algorithms do well, we conduct these algorithms and recommend movies to every user. And we will analyze the recommended movie list. We define favorite movies for certain user as movies with rating higher than 2. After analyzing the test set, we know that every user has 6 favorite movies averagely. So, firstly we use item-based collaborative filtering , Latent factor model, Genre correlation-based recommendation as well as two hybrid methods to recommend 6 movies to every user. And then employ the precision, recall, F-score and coverage metrics which are widely used in movie recommender systems to evaluate intelligence level of recommendations.

Precision is the ratio of favorite movies retrieved by recommendation method to the number of recommendations. Recall gives the ratio of favorite movies retrieved that is considered interesting. F-score is a comprehensive evaluation metrics combine precision and recall. Here we are more concerned about the precision, so precision has higher weight. Coverage is the total number of recommended movies as a percentage of all movies. High coverage means the algorithm will recommend more kinds of movies to user. The evaluation metrics are calculated from the following’s equations:



Fig. 3. Precision, Recall, F-score and coverage on different neighbor size and similarity metrics for ICF.

(6)

(7)

(8)

(9)

## Experiment Design

To compare all the algorithms, we firstly implement ICF, LFM Genre correlation-base algorithm and see the result. Then we will combine these three algorithms to build two hybrid methods and also compute the precision and coverage to evaluate the performance.

For item-based collaborative filtering, we implement it with different number of neighbors and similarity metric. These similarity metrics are used to find similar neighbors: Pearson similarity, Jaccard similarity and Inverse user frequency(IUF). Pearson similarity and Jaccard similarity is popular and powerful similarity metrics in machine learning. Inverse user frequency compute similarity by reducing the impact of active users on similarity calculations. We also combine time effect with Jaccard and IUF to build model so that the movies rated by user long time ago will have less impact than movies rated recently. And when compute the recommend point p(u,i), we will add time effect in it. The metrics are calculated from the following’s equations, where N(i) is set of users who have rated movie I, N(u) is set of movies rated by user u, t is the time of rating [7].

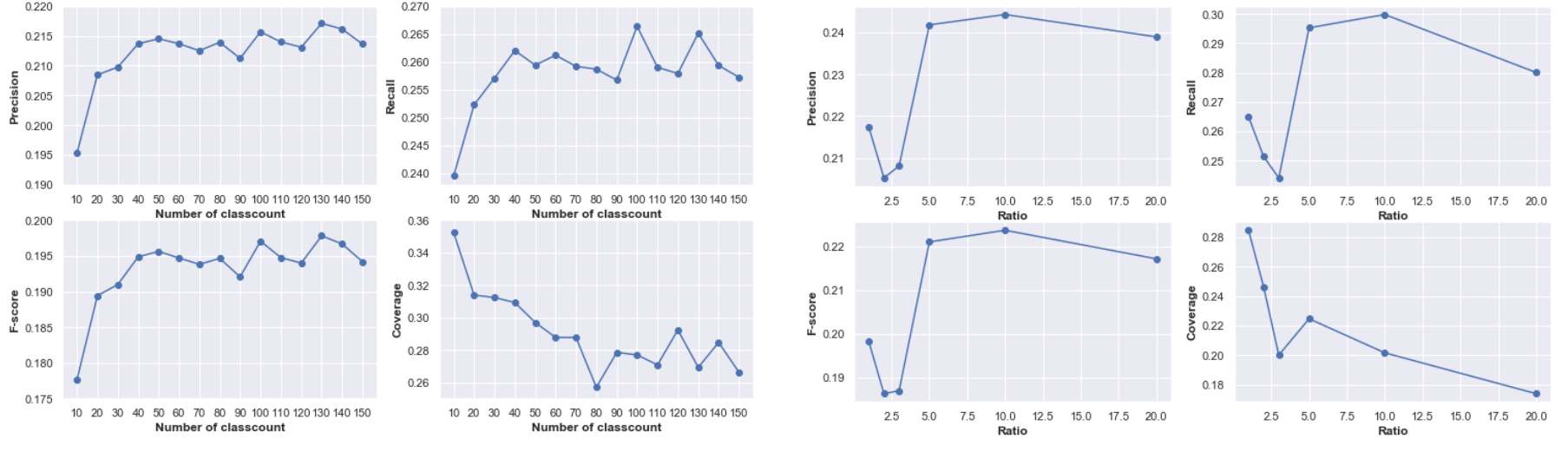


Fig. 4. Precision, Recall, F-score and coverage on different classcount size and ratio for LFM

(10)

(11)

(12)

(13)

For latent factor model, we firstly use 5-cross validation to find the best learning rate(0.015)and penalty parameter(0.005), then try different number of classcounts and ratio to train the model. When picking the positive items, we use all user’s favorite movies. For the negative items, we generate item pools by putting all rating data in it, then randomly pick negative items form the item pools such that popular movies are more possible to be selected. For Genre correlation-based recommendation, we just implement it and compute the precision and recall.

For the hybrid method, we try two different methods to combine these algorithms. The first hybrid method is composed of item-based collaborative filtering and latent factor model. Firstly, we analyze the result of item-based collaborative filtering and latent factor model, then we implement cross validation to find the best parameters for these two models. We use these parameters to train ICF and LFM models and add them up with weights to build the first model. The second hybrid model is implemented by getting one recommended movie from Genre correlation-based algorithm and two movies from the first hybrid model. Two hybrid methods are conduct and calculate the precision and coverage.

We will not design experiment for Top list algorithm because it is quite straightforward, and it can be used to recommend movies if we know nothing about users.

## Result And Discussion

In our experiments, we compute precision and coverage for all models and calculate recall and F-score for some algorithms. But we will pay more attention on the precision and coverage. Table 1 shows the precision and coverage of different algorithms with certain parameters. We will introduce these parameters in hybrid model part.

### Performance of Item-based Collaborative Filtering

We first try to evaluate the performance of Item-based collaborative filtering. Fig. 4 shows the Precision, Recall, F-score and Coverage of ICF with different number of neighbors and similarity metrics. With the neighbor size increase from 10 to 60, precision, recall and F-score increase while coverage decrease. But if neighbor size is bigger than 60, they will all decrease. That is because if we take too many neighbors into consideration, some unsimilar items will also be selected and that will influence the performance of models. So, choosing adequate neighbor size has a big impact on the model. Models with Inverse user frequency have higher precision than model with other similarity metrics, which means that it is reasonable to increase the impact of inactive users and reduce the impact of active users. ICF-IUF-time model performs apparently high precision among all the algorithms, and produces the highest recall, F-score values continually where the neighbor size varies. All models generate decreasing coverage and some models with low precision, such as Pearson model, have high coverage than other. It indicates the coverage of recommendation will be influenced by the quality of model. When models can predict what users like precisely, they will only recommend some hot and good movie, but not recommend randomly. The range of movies that may be recommended will decrease if the model become more precise.

### Performance of Latent Factor Model

Fig 5 shows the result of implementing Latent Factor Model with different classcount size and ratio. Overall, when the classcount size rises, the precision, recall, F-score will also rise, and coverage will decrease. However, we can see the downward trend of precision when the classount size is greater than 130. Because the model overfit if we assume to many classes for user and movie. It is similar to the polynomial regression problem. The model will underfit if the degree is too small, and it will overfit if the degree is too great. Models with ratio equal to 10 perform better than any other models. High ratio means more training data and should produce better predictor, but it leads to bad performance. It due to our definition for negative sample. We consider movies as negative samples if user have not rated it. However, the important thing is that the user didn't rate the movie probably because he hasn't found the movie

yet, instead of he doesn't like the movie. When we take more negative samples, we are more likely to get this type of movie and bring wrong information to the model. The value of hyperparameter will influence Latent Factor model greatly. LFM has lower precision and coverage than ICF-IUF-time.

### Performance of Genre correlation-based algorithm

We train Genre correlation-based model and compute the precision, coverage of recommended list. The precision is 7.2% and the coverage is 30%. The number indicates that Genre correlation-based algorithm cannot make precise prediction. It is reasonable because this algorithm does recommendation only based on user preference and movies genres, so the precision should not be high. The purpose of this algorithm is tackling the cold start problem. Other algorithms are more recommended if we have more information about user.

### Performance of Hybrid method

To generate hybrid models, we observe the result of Item-based Collaborative Filtering and Latent Factor model. Then we choose neighbor size = 50, and to train ICF-IUF-time model(Precision: 26.0% ,Coverage= 42.7%). And choose Classcount=100, itercount = 30, learning rate = 0.015, lambda = 0.005, ratio=10 to train LFM model(Precision: 24.4%, Coverage = 20.1%). Then we build the first hybrid by adding the two models with weights 0.4 and 0.6 respectively. The precision of the first hybrid model increase slightly to 26.7% while the coverage is 38.2%. The result shows that by combining individual recommend algorithm, it is possible to generate more precise hybrid method. However, the trend of accuracy and coverage is often the opposite. The second hybrid model is built by taking five movies from the first hybrid model and one movie from Genre correlation-based model. The precision and coverage are 24.3% and 43.7%. We get a model with larger coverage than any other we implemented before, but it shows lower precision comparing to other models.

# Conclusion

In this paper we implement three recommendation algorithms: Item-based collaborative filtering, Latent Factor Model and Genre correlation-based model. Then we build two hybrid methods by combining these three models. The result tells that in the sparse data environment, find selection of “like-minded” neighborhood on the basis of common ratings is a vital function to generate high quality movie recommendations. Inverse user frequency with time effect is a efficient metrics to find neighborhoods, so ICF-IUF-time is the most precise methods. Latent Factor Model does not work better than ICF with this sparse dataset due to the negative samples we pick may bring wrong information to the models. Hence, LFM is more recommended if we have dataset that is less sparse. Genre correlation-based algorithm tackle the cold start problem, it gives a direction of recommending movies to new user. Two hybrid models show that we can control the precision and coverage by combing individual algorithm in different ways. So far, we conclude from the results that accuracy and coverage always develop in the opposite direction. High precision leads to small coverage and large coverage leads to low precision. To determine which algorithms are better, we should take the system into consideration. If the system focuses more on personalized precise recommendation, algorithms generate high precision are adequate. If it is more concerned about diversified recommendation, algorithms with high coverage should be picked.

As for future work, there are two aspects we can improve. The first is the dataset. In most of the case the dataset will be sparse due to the ratio of users and items. This limits the performance of many algorithms such as LFM. So, there are some technique to predict the missing data and solve the sparsity of dataset. It is one way to increase the performance of algorithms. The second is find more information about users and items. It is hard to make precise only depend on the user's historical behavior. If we know other information about the movies such as the concrete actors, directors, or we acquire social information of users, we can implement more algorithm and build more powerful recommend system. So, we will continue to improve the performance by solving sparsity of data and try more algorithm.

# References

[1] Katarya, R. (2018). Movie recommender system with metaheuristic artificial bee. The Natural Computing Applications, 1983–1990.

[2] Rahul Katarya, O. P. (2017). An effective collaborative movie recommender system with cuckoo. Egyptian Informatics Journal 18, 105-112.

[3] Sang-Min Choi, S.-K. K.-S. (2012). A movie recommendation algorithm based on genre correlations. Expert Systems with Applications, 8079-8085.

[4] Sujoy Bag, S. K. (2019). An efficient recommendation generation using relevant Jaccard similarity. Information Sciences, 53-64.

[5] Francesco Ricci, L. R. (2010). Recommender Systems Handbook. Boston,MA: Springer, 114-129, 151-153.

[6] Zan Wang, X. Y. (2014). An improved collaborative movie recommendation system using computational intelligence. Journal of Visual Languages and Computing, 667-675.

[7]Liang Xiang. (2012). Recommended system practice[M]. POSTS&TELECOM PRESS, 51-57, 131.

1. [↑](#footnote-ref-1)