Hindawi
Scientific Programming
Volume 2021, Article ID 2592604, 14 pages
https://doi.org/10.1155/2021/2592604

Research Article

Novel Multidimensional Collaborative Filtering Algorithm Based on Improved Item Rating Prediction

Tongyan Li,[1] **Yingxiang Li**,[1] **and Chen Yi-Ping Phoebe**[2]

[1]*Chengdu University of Information Technology, Department of Communication Engineering, Chengdu 610225, China*
[2]*Department of Computer Science and Information Technology, La Trobe University, Melbourne, VIC 3086, Australia*

Correspondence should be addressed to Tongyan Li; sunny_cs061@163.com

Received 26 August 2021; Accepted 9 October 2021; Published 5 November 2021

Academic Editor: Punit Gupta

Current data has the characteristics of complexity and low information density, which can be called the information sparse data. However, a large amount of data makes it difficult to analyse sparse data with traditional collaborative filtering recommendation algorithms, which may lead to low accuracy. Meanwhile, the complexity of data means that the recommended environment is affected by multiple dimensional factors. In order to solve these problems efficiently, our paper proposes a multidimensional collaborative filtering algorithm based on improved item rating prediction. The algorithm considers a variety of factors that affect user ratings; then, it uses the penalty to account for users' popularity to calculate the degree of similarity between users and cross-iterative bi-clustering for the user scoring matrix to take into account changes in user's preferences and improves on the traditional item rating prediction algorithm, which considers user ratings according to multidimensional factors. In this algorithm, the introduction of systematic error factors based on statistical learning improves the accuracy of rating prediction, and the multidimensional method can solve data sparsity problems, enabling the strongest relevant dimension influencing factors with association rules to be found. The experiment results show that the proposed algorithm has the advantages of smaller recommendation error and higher recommendation accuracy.

1. Introduction

Recommendation algorithms are mainly divided into six categories: content-based filtering, collaborative filtering, recommendation based on association rules, recommendation based on utility, recommendation based on knowledge, and mixed recommendation [1, 2]. Collaborative filtering (CF) algorithms are the most widely used and classic because of their easy implementation, high accuracy, and high recommendation efficiency. However, in the era of big data, one typical feature is that the amount of data is huge but the information density is low, which can also be called information sparse data. Collaborative filtering algorithms are often ineffective when dealing with large amounts of sparse data. Furthermore, the complex data environment results in many factors affecting the recommendation. With the development of the mobile Internet, mobile devices can easily obtain more information about dimensions, such as location, weather, and social relationships. Under different external influences, the recommendation results will change greatly. However, most of the current collaborative filtering algorithms are based on a single dimension for recommendation.

In order to improve the performance of collaborative filtering recommendation algorithms, researchers resolve the problems from different perspectives and propose a variety of recommendation algorithms. Some researchers optimized the user scoring matrix using different methods [3–6], and others used fuzzy sets to efficiently represent user features [7, 8]. These methods all effectively alleviate the problem of sparse data. In order to find a neighbor set that is more similar to the target user's interest and improve the accuracy of recommendations, some researchers improved the similarity calculation method [7, 9, 10], and others used location information and trust relationship information, such as [11, 12]. The potential relationship between

information mining users provides new ideas for finding neighbors. Researchers have also used demographic knowledge [13, 14] to achieve major breakthroughs, while some scholars used score ranking prediction methods to enhance recommendation performance such as [15–17], and others chose the genetic algorithm used in the prediction process to improve recommendation performance, such as [18, 19]. Context as a dynamic description of an item and a user's situation affects the user's decision-making process; hence, it is essential for any recommendation system in a big data environment [20–22].

These algorithms alleviate the problems caused by data sparsity to some extent, improve the accuracy of calculation similarity and recommendation quality with different methods, but the implementation of the algorithm depends on a large amount of user information and calculation, which can be due to high complexity and hard implementation.

In this paper, we focus on the recommendation algorithm of data in complex environments. Firstly, we study the traditional item rating prediction algorithm and make some improvements with adding the weight problem considering user score. We introduce the system error factor based on statistical learning for the user to develop a personalized rating prediction algorithm. Then, we propose a personalized rating prediction method that is combined with a classical collaborative filtering algorithm User-Inverse Item Frequency (User-IIF) [23] to develop a novel collaborative filtering algorithm. This method is based on both User-IIF and personalized rating prediction. Secondly, we focus on the impact of multidimensional factors and propose a novel multidimensional method, which can separate user groups based on context-aware dimensions combined with both user clustering and item clustering. Finally, we conduct a series of experiments to prove that our algorithms and methods are effective and efficient. The experiment results prove that our algorithms are easy to implement with low computational overhead. In addition, our algorithms can also process sparse data and improve the accuracy of recommendations.

The rest of the paper is organized as follows. The works related to our research and our novel methods to deal with multidimensional recommendation are proposed in Section 2, and the experiment results and discussion are presented in Section 3. Finally, the conclusion and suggestions for future work are given in Section 4.

2. Methodology

2.1. Proposed Item Rating Prediction Method. Traditional item rating prediction algorithms take the target user's average item historical score as the reference center and then use the similarity between similar neighbors to perform item rating prediction. When user data is sparse, the error rate of traditional item rating prediction algorithms increases and the accuracy rate decreases. This proposed method adds weights to consider user ratings and introduces systematic error factors based on statistical learning to improve traditional item rating prediction algorithms.

2.1.1. User Rating Weighting Factor. Traditional item rating prediction algorithms take the historical average score of the target user as the central value and rely on the neighbor's score to correct it. Traditional algorithms rely too much on the user's score and its anti-interference ability is ineffective when it is faced with data sparseness. For example, when a user has not scored many items, even if the average user score is close to 0, using the user's historical average score as the center value may result in inaccurate recommendation results. The item scoring prediction method proposed in this work considers the factors of public scoring, improves the algorithm using (1), introduces the weighting factor a of the user's score, and assigns the weight of the scoring $(1-\alpha)$ to the scoring of the item.

$$r_{ui} = a \cdot \overline{r_u} + (1-a) \cdot \overline{r_i} + \frac{\sum_{v \in s(u,K) \cap N(i)} \mathrm{sim}_{uv} \left(r_{vi} - \overline{r_v}\right)}{\sum_{v \in s(u,K) \cap N(i)} \left|\mathrm{sim}_{uv}\right|}. \tag{1}$$

2.1.2. Systematic Error Factor Based on Statistical Learning. A large number of studies show that there are errors in item rating prediction, which only a few algorithms have addressed by performing a statistical analysis calculation on each recommendation result. In order to achieve more personalized recommendations, it is necessary to establish a system error factor generated by the recommendation system for each user.

The system error ε_u generated by the recommendation of target user u is calculated by (2), where the actual score of user u on item i is represented as R_{ui}, and r_{ui} describes the predicted rating of user u generated by the system. $N_{I(u)}$ describes the number of items in the itemsets $I(u)$ that target user u adopts from the recommendation results. Through statistical learning, the system sets the error factor for each user, and then this is applied to the collaborative filtering algorithm to correct the item rating prediction, as shown in (3), for a more accurate personalized recommendation algorithm for the target user.

$$\varepsilon_u = \frac{\sum_{i \in I(u)} \left(R_{ui} - r_{ui}\right)}{N_{I(u)}}, \tag{2}$$

$$r_{ui} = a \cdot \overline{r_u} + (1-a) \cdot \overline{r_i} + \frac{\sum_{v \in s(u,K) \cap N(i)} \mathrm{sim}_{uv} \left(r_{vi} - \overline{r_v}\right)}{\sum_{v \in s(u,K) \cap N(i)} \left|\mathrm{sim}_{uv}\right|} + \varepsilon_u. \tag{3}$$

Based on the above calculations, this paper proposes a novel algorithm, namely, improved item-rating prediction (IIP) for user scoring. The main steps of IIP are shown in Figure 1(a). The basic idea is to form a set of error factors for

ALGORITHM 1: IIP (Improved Item-rating Prediction)

Input: (1) R, user rating matrix (R[user][item]=rating);
(2) U_R, user's average historical rating (U_R[user]=rating);
(3) I_R, average rating of item (I_R[item]=rating);
(4) S, user similarity matrix (S[user][sim_user]=sim);
(5) A, user rating weighting factor (0<A<=1);
Output: P_R, predicted item-rating (P_R[user][item]=rating);

```
01. For each user in R do
02.  { N[user]←Number of items that user like;
03.    E[user]←The sum of the difference between the predicted rating and the true;
04.    AVGE[user]←E[user] / N[user];
05.    Set fz←Ø; fm←Ø;
06.    For each sim_user, sim in Top-K S[user] do
07.    { For each item in R[sim_user] do
08.      { If item not in R[user] then
09.        { fz.setdefault(item, 0.0); fm.setdefault(item, 0.0);
10.          fz[item]←fz[item]+sim*(R[sim_user][item])-U_R[sim_user]);
11.          fm[item]←fm[item]+sim;
12.        }
13.      }
14.    }
15.    For each item in fm do
16.     P_R[user][item]←A*U_R[user]+(1-A)*I_R[item]+fz[item]/fm[item]+AVGE[user];
17.  }
```

(a)

ALGORITHM 2: cross-iterative biclustering

Input: (1) U, user set;
(2) I, item set;
Output: Adjusted cluster;

```
01. By K-means clustering method, get the initial user cluster UC ={uc1, uc2, ⋯, uck) and clustering center UCC ={ucc1, ucc2, ⋯, ucck), the initial item cluster IC= {ic1, ic2, ⋯, ick) and clustering center ICC ={icc1, icc2, ⋯, icck );
02. Repeat;
03. For each user cluster uci∈UC do
04.  { For each user uj∈uci do
05.    { Calculat the correlation SU(uj, ucci) between uj and clustering center ucci;
06.      If (SU(uj, ucci) > ε)
07.       continue;
08.      else
09.       Calculate the similarity between uj and other cluster centers, and add it to the cluster with the most similarity;
10.    }
11.  }
12.  Recalculate the center of each cluster in UC and update to UCC;
13.  For each item cluster ici∈IC do
14.  { For each item ij∈ici do
15.    { Calculat the corr elation SI(ij, icci) between ij and clus tering center icci;
16.      If (SI(ij, icci) > η)
17.       continue;
18.      else
19.       Calculate the similarity between ij and other cluster centers, and add it to the cluster with the most similarity;
20.    }
21.  }
22.  Recalculate the center of each cluster in IC and update to ICC;
23.  Until clusters, the elements in the cluster are no longer separated or reach the set number of iterations.
```

(b)

ALGORITHM 3: Context similarity calculation

Input: (1) I_R, average rating of item (I_R[item]=rating);
(2) I, user rating and corresponding context information;
(3) contexts X and Y;
(4) current context t;
Output: S_{xyi}, similarity between contexts X and Y;

```
01. Xt←Standard deviation of context dimension X;
02. Yt←Standard deviation of context dimension Y;
03. f←Ø;
04. For each user, item in I do
05.  { f.setdefault(item, 0.0);
06.    f←f+(Ixt[user][item]-I_R[item])*(Iyt[user][item]-I_R[item]);
07.  }
08. Sxyi=f/( Xt* Yt);
```

(c)

ALGORITHM 4: Multi-dimensional context-aware

Input: (1) user, target user;
(2) I, user rating and corresponding context information;
(3) U_R, user's average historical rating (U_R[user]=rating);
(4) S, user similarity matrix (S[user][sim_user]=sim);
Output: P_R, predicted item-rating (P_R[item]=rating);

```
01. For each u, i in I do
02.  // Apply the context similarity algorithm to calculate S(c,x,t);
03.  R[u][i] ←
04. Set fz←Ø; fm←Ø;
05. For each sim_user, sim in Top-K S[user] do
06. { For each item in R[sim_user] do
07.   { If item not in R[user] then
08.     { fz.setdefault(item, 0.0); fm.setdefault(item, 0.0);
09.       fz[item]←fz[item]+sim*(R[sim_user][item])-U_R[sim_user]);
10.       fm[item]←fm[item]+sim;
11.     }
12.   }
13.  }
14.  For each item in fm do
15.   P_R[item]←U_R[user]+fz[item]/fm[item];
16. }
```

(d)

Figure 1: Multidimensional collaborative filtering algorithm based on improved item rating prediction. (a) Algorithm 1. (b) Algorithm 2. (c) Algorithm 3. (d) Algorithm 4.

each user through statistical learning, and then apply it to the collaborative filtering algorithm to correct the item rating prediction.

2.2. User Scoring Based on Multidimensional Context

2.2.1. User Similarity Calculation. The first step of our method is to get the user's neighbor cluster, which is obtained through the user's scoring matrix. Users in the user group whose interests are similar to each other can be selected as neighbor users. This paper utilizes Pearson's similarity [24] to measure the distance between users as shown in (4). Pearson's similarity is similar to cosine similarity in form. The average evaluation value of users is subtracted during calculation, which is to normalize the cosine similarity and unify the user's scoring standard. The range of Pearson's similarity is [−1, 1], which is more accurate than that of Jaccard's correlation coefficient and cosine similarity.

$$s_{u,v} = \frac{\sum_{i \in I_u \cap I_v} \left(r_{u,i} - \overline{r_u}\right)\left(r_{v,i} - \overline{r_v}\right)}{\sqrt{\sum_{i \in I_u \cap I_v} \left(r_{u,i} - \overline{r_u}\right)^2 \sum_{i \in I_u \cap I_v} \left(r_{v,i} - \overline{r_v}\right)^2}}, \tag{4}$$

where $s_{u,v}$ represents the similarity value between target user a and its neighbor cluster user, I_u represents the set of items that target user u has scored, and I_v indicates the set of products scored by neighbor cluster user v. i represents the item that the target user u and neighbor cluster user v scored together. $r_{u,i}$ indicates the rating of item i by target user u, and $\overline{r_u}$ indicates the average rating of target user u. Following the same principle, $r_{v,i}$ is the score of neighbor cluster user v for item i, and $\overline{r_v}$ indicates the average rating of neighbor cluster user v. The traditional collaborative filtering algorithm uses the above formula to calculate the similarity between users.

A user will have different scoring standards under different contexts, such as the user's rating of a hotel when traveling on business and the rating criteria for a hotel when traveling privately. So after considering the context, it has nothing to do with the previous rating and is replaced by a new symbol, we use $\overline{r_{u,c}}$ instead of $\overline{r_u}$ and $\overline{r_{v,c}}$ instead of $\overline{r_v}$, where $\overline{r_{u,c}}$ represents the average rating of user u under context condition c. The range of c can be appropriately generalized or filtered as needed. The proposed improved method can be described as follows:

$$s_{u,v,c} = \frac{\sum_{i \in I_u \cap I_v} \left(r_{u,i} - \overline{r_{u,c}}\right)\left(r_{v,i} - \overline{r_{v,c}}\right)}{\sqrt{\sum_{i \in I_u \cap I_v} \left(r_{u,i} - \overline{r_{u,c}}\right)^2 \sum_{i \in I_u \cap I_v} \left(r_{v,i} - \overline{r_{v,c}}\right)^2}}. \tag{5}$$

The improved user neighbor cluster similarity calculation formula takes into account the influence of context factors on the basic rating, making the calculation formula closer to the context recommendation environment. After considering the context, the user's similarity calculation (4) is improved to (5), and the influence of the context on the user's rating is taken into account when using the mean calibration error of the score.

2.2.2. User and Item Cross-Iterative Bi-Clustering. The cross-iterative bi-clustering method is used for cluster users and items separately. Due to the sparsity of the user-item matrix, the initial clustering is not accurate enough. Therefore, we use the cross-iterative method to adjust both user clustering and item clustering.

User clustering adjustment is calculated by (6), and item clustering adjustment is calculated by (7)

$$S_U(u_t, u_c) = \begin{cases} 1, & u_t = u_c, \\ \dfrac{1}{\sum_{k=1}^{n} r_{tk} \sum_{k=1}^{n} r_{ck}} \displaystyle\sum_{i_i \in I(u_t, u_c)} \sum_{i_j \in I(u_t, u_c)} \operatorname{sim}\left(i_i, i_j\right), & \text{else}, \end{cases} \tag{6}$$

where r_{tk} is the score of user u_t for each item and r_{ck} is the score of user u_c for each item. $I(u_t, u_c)$ is a collection of items that u_t and u_c have scored together, and $\operatorname{sim}(i_i, j_j)$ is the similarity between items i_i and j_j. Here, we also use Pearson's similarity to calculate this. If there are a lot of items that have been rated together, they can be considered as users with similar interests. If the obtained $S_u(u_t, u_c)$ is greater than a certain threshold ε, it can be kept in the cluster; otherwise, it will be separated from the current cluster. Then, we calculate the similarity between u_t and the other cluster centers. This is added to the cluster with the most similarity to complete the adjustment of the user cluster.

$$S_I(i_t, i_c) = \begin{cases} 1, & i_t = i_c, \\ \dfrac{1}{\sum_{k=1}^{m} r_{tk} \sum_{k=1}^{m} r_{ck}} \displaystyle\sum_{u_i \in U(i_t, i_c)} \sum_{u_j \in U(i_t, i_c)} \operatorname{sim}\left(u_i, u_j\right), & \text{else}, \end{cases} \tag{7}$$

where r_{tk} is the score of each user on item i_t and r_{ck} is the score of each user on item i_c. $U(i_t, i_c)$ is the set of users that i_t and i_c have scored together. $\mathrm{sim}(u_i, u_j)$ is the similarity between items u_i and u_j. Here, we use Pearson's similarity. If the obtained $S_I(i_t, i_c)$ is greater than a certain threshold η, the item will be kept in the cluster; otherwise, it will be separated from the current cluster. Then, we calculate the similarity between i_t and the other cluster centers. This is added to the cluster with the most similarity to complete the adjustment of the user cluster. Algorithm 2 is proposed for cross-iterative bi-clustering, as shown in Figure 1(b).

2.2.3. Context Similarity Calculation. When the scope of the context is very large, there are many different dimensions, such as time, place, surrounding people, etc. According to the characteristics of the dataset and the environment collection ability, the context dimension selected by the recommendation system will be different. As far as the time dimension is concerned, it can also be specifically subdivided into seasons, weeks, moments, holidays, and so on.

Assume that we select a system with z different dimensions, which is shown as $c = (c_1, c_2, \ldots, c_z)$, where $c_t (t = 1, \ldots, z)$ is a contextual dimension (such as time, location, weather, etc.). The similarity of the context between two score records x, y on dimension t can be recorded as $\mathrm{sim}_t(x, y)$. We use the degree of influence of the context dimension on the score to measure the similarity between the two context variables as follows:

$$\mathrm{sim}_t(x, y, i) = \frac{\sum_{u=1}^{n}\left(r_{u,i,x_t} - \overline{r_i}\right)\cdot\left(r_{u,i,y_t} - \overline{r_i}\right)}{\sigma_{x_t}\cdot\sigma_{y_t}}, \tag{8}$$

where u is the user and r_{u,i,x_t} describes the rating of item i under the context of x_t by user u. $\overline{r_i}$ is the average score of item i. Similarly, r_{u,i,y_t} is user u's rating of item i under context y_t, σ_{x_t} is the standard deviation of context dimension x_t, and σ_{y_t} is the standard deviation of context dimension y_t. This paper proposes a novel method to measure the similarity of context x and y, according to the influence degree of different contexts on the score of the same commodity i in the t dimension. Algorithm 3 is shown in Figure 1(c) to calculate context similarity efficiently.

2.2.4. The Proposed Multidimensional Context-Aware Based Method. In multidimensional recommendation, the addition of context results in a lot of interesting rules and mining high-frequency patterns between contexts and items can help discover the impact of different contexts on user decisions. In this paper, we select the multidimensional context from strong association rules with the algorithm FP-growth.

Generally, when determining the neighbor user group, the N-user with the largest similarity can be selected as the cluster neighbor of the target user according to the similarity calculation formula. The context can help the user to filter out some of the user score records that have a large difference in context from the current recommendation environment. Because some commodity decisions are closely related to a certain context factor, the context is called a hard context and must be considered and satisfied in the recommendation. Some score records that do not satisfy the current context can be filtered out preferentially and are not considered when calculating the similarity of neighbor clusters.

Due to the influence of the context, the user's rating record has its own context background, and the target user's current background is different, so the rating record in different contexts is different from the user in the current context. In order to distinguish the relevance of the rating record under the current context, we use the contextual similarity calculation method to calculate $\mathrm{sim}_t(x, y, i)$, which describes the similarity between context x and context y in the t dimension. The user rating predictions in a multidimensional context can be described as follows:

$$r_{u,i,c} = \alpha\cdot\overline{r_u} + \beta\cdot\overline{r_i} + (1-\alpha-\beta)\cdot\overline{r_c} + \frac{\sum_{v\in S(u,K)\cap N(i)}\mathrm{sim}_{u,v}\left(r_{v,i} - \overline{r_v}\right)}{\sum_{v\in S(u,K)\cap N(i)}\left|\mathrm{sim}_{u,v}\right|} + \varepsilon_u, \tag{9}$$

where c is the context in which the target user is located and ε_u is the system error (the other symbols are described in the previous formula). It is well known that contexts can have many specific dimensions, depending on the data collection, such as time, location, and related personnel. The time dimension can be divided into seasons, weeks, moments, holidays, and so on.

After comprehensively considering the influence of context on the recommendation system, (10) can be replaced with (11). The basic clustering rating prediction formula is modified as follows:

$$R_{u,i,c} = k\sum_{x\in c}\sum_{c=1}^{z} r_{u,i,c}\cdot\mathrm{sim}_t(x, y, c), \tag{10}$$

$$P = \overline{r_{u,c}} + \frac{\sum_{v\in N_c}\left(R_{v,i,c} - \overline{r_{v,c}}\right)\times\mathrm{sim}(u, v, c)}{\sum_{v\in N_c}\mathrm{sim}(u, v, c)}. \tag{11}$$

Algorithm 4 shown in Figure 1(d) is proposed as the multidimensional context-aware based method. Using this algorithm, item scores can be obtained under multidimensional conditions.

3. Experiments and Results

3.1. Experimental Datasets and Environment. In order to verify the impact of a user's scoring weight on the recommendation results and to prove that the recommendation accuracy of the collaborative filtering algorithm based on the user and improved item scoring is more accurate, it is necessary to compare our proposed algorithm with traditional algorithms that are based on classical item scoring prediction methods.

These experiments were conducted under the following conditions: (1) CPU dual core i7-8750H with frequency 2.5 GHz; (2) main memory of 8 G; (3) Windows 10 64-bit operating system; (4) database software version MySQL 5.7. The proposed machine learning algorithms are implemented

using an object-oriented dynamic type interpreted scripting language Python, including Python itself with some powerful libraries and third-party modules which cover scientific computing, database interfaces, etc., such as NumPy, pandas, etc. The integrated development environment is JetBrains PyCharm Professional 2018.2.5.

The experiments to improve user rating prediction utilize two datasets, MovieLens and Jester. The Jester dataset was developed by Ken Goldberg and his team at the Berkeley University of California. The Jesters dataset scored [−10, 10], the jester-dataset-1 comprises data from 24,983 users who have rated 36 or more jokes, a matrix of 24983×101; and jester-dataset-3 comprises data from 24,938 users who have rated between 15 and 35 jokes, a matrix of 24938×101 dimensions. The MovieLens dataset was organized by the Group Lens team at the University of Minnesota. The MovieLens 100K dataset comprises 100,000 ratings from 1000 users for 1,682 movies, rated between 1 and 5. The sparsity of the set is about 93.7%, and the data sparse problem is evident. In the experiment, in order to simulate datasets of different scales and different sparsity levels, the existing datasets were processed to generate four datasets as shown in Table 1. From this table, we can see that the datasets were randomly divided into training sets and test sets during the experiment, where the training set accounted for 80% of the entire dataset, and the test set accounted for the remaining 20%. When the Jesters dataset was processed, the score was formulated with the value 0 to 5 by $r^* = (r + 10)/4$.

The source of the experimental datasets for testing the multidimensional collaborative filtering algorithm is CARSKit (https://github.com/irecsys/CARSKit/), which is an open-source Java-based context-aware recommendation engine. We used two datasets: DePaulMovie [25] and TripAdvisor_v1 [26]. In the experiments, DePaulMovie kept its original shape, and TripAdvisor_v1 was filtered and adjusted. We used 70% of the dataset as the training set and 30% as the test set.

3.2. Evaluation Indicators of the Experiment Results. In order to study the performance of the improved recommendation algorithm, the experiment used four indicators, namely, precision, recall, mean absolute error (MAE), and root mean square error (RMSE).

Precision is an important indicator for evaluating the performance of a recommendation algorithm. It describes the proportion of the recommended items that the recommendation system makes for the user. The larger its value, the higher the accuracy of the system, and the better the system's recommendation. Precision is computed as shown in the following formula:

$$\text{precison} = \frac{\sum_u |R(u) \cap T(u)|}{\sum_u |R(u)|}, \tag{12}$$

where u is the target user who uses the system, $R(u)$ is the set of recommended items for the user, and $T(u)$ is the set of items in which the user is actually interested.

"Recall" describes how many of the products the user is interested in and how many are actually recommended to him by the system. Recall is computed as shown in the following formula:

$$\text{Recall} = \frac{\sum_u |R(u) \cap T(u)|}{\sum_u |T(u)|}. \tag{13}$$

The molecular weight of recall is the same as the molecule of the precision, which is the intersection of $R(u)$ and $T(u)$; however, their denominators are different. The denominator part of the accuracy rate is $R(u)$, which is the set of all items recommended to the user, and the denominator of the recall rate is $T(u)$, which is the collection of all the items of interest of the user. A larger recall corresponds to a better performance.

MAE avoids the problem where the errors cancel each other out and accurately reflects the actual prediction error. The calculation method is as shown as in formula (14), which averages the absolute value of the difference between the actual score and the predicted score.

$$\text{MAE} = \frac{1}{m} \sum_{i=1}^{m} |Y_i - y_i|, \tag{14}$$

where Y_i and y_i denote the original data and predicted data, respectively.

RMSE is used to measure the deviation between the observed value and the true value. The calculation method is shown in formula (15), that is, the ratio of the square of the difference between the predicted score and the actual score to the m number of observations squared.

$$\text{RMSE} = \sqrt{\frac{1}{m} \sum_{i=1}^{m} (Y_i - y_i)^2}. \tag{15}$$

The smaller the MAE and RMSE values, the better the recommended performance of the algorithm.

3.3. Experiment Results

3.3.1. Choosing the Best Value for the User Score Weighting Factor. Firstly, the optimal value for the user's score weighting factor a is determined for the collaborative filtering algorithm (U&IPRP-CF), based on the user and the improved item rating prediction. To ensure the accuracy of the experiment, the number of item recommendations N of the Jester-500-100, Jester-1000-100, and Jester-1000-200 datasets is set to a constant $N = 10$, meaning that 10 items are recommended to each target user. However, the number of items in the MovieLens dataset is large, and the number of recommended items N is set to a constant $N = 30$, meaning that 30 items are recommended to each target user. The K number of the most similar neighbors selected for each target user is a variable, and K is taken from 50, in increments of 10, and sequentially taken to 100, that is, [50, 60, 70, 80, 90, 100].

Each dataset was tested on a set of values of $a = 1, 0.9, 0.8, 0.7, 0.6, 0.5$ for a. The experiment results of the Jester-500-

Table 1: Dataset and related parameters.

Item	User number	Item number	Score	Sparseness
Jester-500-100	500	100	35,861	0.283
Jester-1000-100	1000	100	70,675	0.293
Jester-1000-200	1000	200	50,516	0.747
MovieLens	943	1,682	100,000	0.937

100 dataset are shown in Tables 2 and 3. The best value of a is around 0.9. The experiment results of the Jester-1000-100 dataset are shown in Tables 4 and 5. The best value of a is around 0.8 and 0.9. The experiment results of the Jester-1000-200 dataset are shown in Tables 6 and 7. The best value of a is around 0.7. The experiment results of the MovieLens dataset are shown in Tables 8 and 9. The best value of a is around 0.7. In summary, as the size of the dataset increases and sparsity increases, the optimal value of the user's score weighting factor a decreases.

After determining the best value of the user's score weighting factor, a comparative experiment is carried out, and the algorithms are first sorted and named, as shown in Table 10.

The algorithms are compared in Table 10. In order to ensure the accuracy of the experiment, the recommended number N of Jester-500-100, Jester-1000-100, and Jester-1000-200 datasets is a constant $N = 10$. The recommended N number of items in the MovieLens dataset is a constant $N = 30$. The K number of most similar neighbors selected for each target user is a variable. K has a value from 50 to 100 with an interval of 10, expressed as [50, 60, 70, 80, 90, 100].

The comparison results of the performance for different datasets are shown in Figure 2. The experiment results for the Jester-500-100 dataset are shown in Figures 2(a) and 2(b). The experiment results for the Jester-1000-100 dataset are shown in Figures 2(c) and 2(d). The experiment results for the Jester-1000-200 dataset are shown in Figures 2(e) and 2(f). The experiment results for the MovieLens dataset are shown in Figures 2(g) and 2(h). It can be seen that for the different datasets, the error generated by the U&IPRP-CF algorithm is smaller than the traditional algorithm, and it has obvious advantages in terms of the accuracy of recommendations.

In the three datasets Jester-500-100, Jester-1000-100, and Jester-1000-200, the U&URWFRP-CF algorithm does not reduce the recommendation error, whereas the recommendation error for the U&URWFRP-CF algorithm is reduced in the MovieLens dataset. This shows that when the dataset is small and sparse, the user's score is very reliable; otherwise, when the dataset is large and information sparse, the user's score is less likely to be referenced under a large base. In this situation, the weights of the user ratings need to be considered.

From these figures, it can be seen that when the scale and sparsity of the dataset are gradually increased, the recommendation error of the U&IPRP-CF algorithm is also reduced and the performance becomes increasingly better, which alleviates the data sparsity problem to some extent.

It can be seen from the experiment results that using the average score of the public to replace the average historical score weight of the user alone can enable the system to predict the user's score on the unrated item, resulting in less error, thus making more accurate recommendations. To reduce the error of score prediction, a systematic error factor is established for each user, and statistical learning is improved on each recommendation, which can effectively correct the error and improve the accuracy of recommendations.

In addition, combined with experiment 3.3.1, it can be seen that choosing the correct parameters is also key to improving the accuracy of recommendations. When the number of items recommended by item N, the number of neighbors of target user K, and the user's score weighting factor a are in a practical application, the recommendation system needs to compare and select appropriate values for the experiment.

3.3.2. Performance Comparison Experiments Using Different Algorithms. The experimental results were verified by DePaulMovie and TripAdvisor_v1 datasets published by GroupLens. Our paper compared four recommendation algorithms: CF, CF-AR, Multi-CF, Multi-CF-AR, CF-AR-IIP, and Multi-CF-AR-IIP, the latter two being the algorithms proposed in this paper. The algorithms are explained as follows:

(a) CF: classical user-based collaborative filtering recommendation algorithm

(b) CF-AR: user-based collaborative filtering recommendation algorithm combined with association rules mining

(c) CF-AR-IIP: user-based collaborative filtering recommendation algorithm combined with association rules mining and improved item-rating prediction, which is proposed in this paper

(d) Multi-CF: classical multidimensional context-aware user-based collaborative filtering algorithm

(e) Multi-CF-AR: multidimensional context-aware user-based collaborative filtering algorithm combined with association rules mining

(f) Multi-CF-AR-IIP: user-based collaborative filtering recommendation algorithm combined with association rules mining and improved item-rating prediction, which is proposed in this paper

In the experiments based on the DePaulMovie datasets, the top-K ($K = 10$, 15, 20, 25) neighbors with the highest similarity for each user and the top-N ($N = 10$, 15, 20) recommended items with the highest predicted rating for the target users are selected. The system has three context

Table 2: Comparison of MAE using Jester-500-100.

a	$K=50$	$K=60$	$K=70$	$K=80$	$K=90$	$K=100$
1	0.5791	0.5781	0.5985	0.5723	0.5539	0.5574
0.9	0.5530	0.5655	0.5591	0.5334	0.5480	0.5675
0.8	0.5411	0.5795	0.5710	0.5350	0.5597	0.5815
0.7	0.5754	0.5729	0.5903	0.5630	0.5356	0.5871
0.6	0.5632	0.5917	0.5917	0.5782	0.5618	0.5746
0.5	0.5747	0.6027	0.5973	0.5806	0.5712	0.5800

Table 3: Comparison of RMSE using Jester-500-100.

a	$K=50$	$K=60$	$K=70$	$K=80$	$K=90$	$K=100$
1	0.8860	0.8982	0.9618	0.9141	0.8662	0.8731
0.9	0.8710	0.8975	0.8942	0.8560	0.8542	0.8966
0.8	0.8529	0.9222	0.9120	0.8507	0.8922	0.9335
0.7	0.9032	0.9243	0.9456	0.8928	0.8469	0.9438
0.6	0.8928	0.9568	0.9396	0.9212	0.8770	0.9056
0.5	0.8976	0.9722	0.9466	0.9487	0.8865	0.9084

Table 4: Comparison of MAE using Jester-1000-100.

a	$K=50$	$K=60$	$K=70$	$K=80$	$K=90$	$K=100$
1	0.5091	0.5105	0.5528	0.5385	0.5199	0.5058
0.9	0.5029	0.5081	0.5366	0.5432	0.5195	0.5225
0.8	0.5006	0.5302	0.5191	0.5263	0.5233	0.5460
0.7	0.5090	0.5141	0.5289	0.5364	0.5377	0.5409
0.6	0.5330	0.5289	0.5572	0.5431	0.5429	0.5337
0.5	0.5417	0.5440	0.5647	0.5534	0.5509	0.5455

Table 5: Comparison of RMSE using Jester-1000-100.

a	$K=50$	$K=60$	$K=70$	$K=80$	$K=90$	$K=100$
1	0.8212	0.8322	0.8932	0.8659	0.8242	0.8212
0.9	0.8120	0.8372	0.8736	0.8865	0.8345	0.8332
0.8	0.8130	0.8713	0.8474	0.8692	0.8418	0.8788
0.7	0.8345	0.8543	0.8753	0.8832	0.8713	0.8722
0.6	0.8783	0.8707	0.9133	0.8792	0.8740	0.8590
0.5	0.8918	0.8885	0.9246	0.8993	0.8822	0.8667

dimensions, namely, $C = (C1, C2, C3) =$ (Time, Location, Companion), where $C1$ (Time) = (Weekday, Weekend), $C2$ (Location) = (Cinema, Home), and $C3$ = (Companion) = (Alone, Family, Partner).

In the experiments based on the TripAdvisor_v1 dataset, the top-K ($K = 20, 30, 40, 50$) neighbors with the highest similarity for each user and the top-N ($N = 10, 15, 20$) recommended items with the highest predicted rating for the target users are selected. The system has three context dimensions as follows: $C = (C1, C2, C3) =$ (USER_TIMEZONE, HOTEL_TIMEZONE, Trip Type), where $C1$ (USER_TIMEZONE) = (Eastern, Central, Pacific, Mountain, HI, AK), $C2$ (HOTEL_TIMEZONE) = (Eastern, Central, Pacific, Mountain), and $C3$ (Trip Type) = (1, 2, 3, 4, 5). Table 11 shows information on the related context dimensions selected for the datasets.

Figures 3(a) and 3(b) show the precision of the algorithms. Multi-CF-AR-IIP achieves the best precision, and Multi-CF-AR is the second best. Particularly when N is small, Multi-CF-AR-IIP has obvious advantages. As N increases, the precision of all the algorithms decreases, and the difference between the algorithms becomes increasingly smaller. This shows that the increase in the number of recommendations reduces the accuracy of the recommendation. Different algorithms will exhibit different characteristics in different datasets. Multi-CF has an advantage in DePaulMovie, but it does not work well in TripAdvisor_v1.

Figures 3(c) and 3(d) show the recall of the algorithms. The result is the same as for precision, where Multi-CF-AR-IIP achieves the best recall, and Multi-CF-AR is the second best. However, recall increases significantly as N increases. This is because the denominator of recall is the number of items in which the user is actually interested and its value is small.

Figures 3(e) and 3(f) show the MAE of the algorithms and Figures 3(g) and 3(h) show the RMSE of the algorithms. In DePaulMovie, except CF, the resulting errors of the other

Table 6: Comparison of MAE using Jester-1000-200.

a	$K=50$	$K=60$	$K=70$	$K=80$	$K=90$	$K=100$
1	0.4139	0.3714	0.2611	0.1727	0.2059	0.1799
0.9	0.3562	0.4590	0.2103	0.2422	0.1879	0.1815
0.8	0.2933	0.3235	0.2426	0.2472	0.1991	0.1999
0.7	0.2309	0.3507	0.1582	0.2160	0.2616	0.2489
0.6	0.3161	0.3401	0.2105	0.2445	0.2566	0.2718
0.5	0.2940	0.3113	0.2763	0.2415	0.2604	0.3213

Table 7: Comparison of RMSE using Jester-1000-200.

a	$K=50$	$K=60$	$K=70$	$K=80$	$K=90$	$K=100$
1	0.8962	0.6459	0.3511	0.3651	0.4679	0.3333
0.9	0.9843	0.5405	0.4266	0.3451	0.4401	0.3403
0.8	0.7912	0.6234	0.4295	0.3887	0.3769	0.3676
0.7	0.7718	0.4226	0.4382	0.5201	0.5442	0.4873
0.6	0.7545	0.5812	0.5336	0.4989	0.5547	0.5323
0.5	0.6692	0.5975	0.5068	0.5343	0.5589	0.6286

Table 8: Comparison of MAE using MovieLens.

a	$K=50$	$K=60$	$K=70$	$K=80$	$K=90$	$K=100$
1	0.2634	0.6377	0.7479	0.7883	0.7851	0.3524
0.9	0.1978	0.4791	0.4692	0.3977	0.0811	0.0249
0.8	0.2178	0.3709	0.3448	0.2918	0.0931	0.1255
0.7	0.2456	0.2994	0.3058	0.2271	0.1942	0.1589
0.6	0.3610	0.3881	0.3552	0.2668	0.1889	0.1499
0.5	0.4612	0.4560	0.3491	0.2803	0.2177	0.2261

Table 9: Comparison of RMSE using MovieLens.

a	$K=50$	$K=60$	$K=70$	$K=80$	$K=90$	$K=100$
1	0.7249	1.4619	1.5619	1.4962	1.5344	0.3524
0.9	0.6457	1.0449	1.0487	1.0665	0.3218	0.0249
0.8	0.6294	0.9029	0.8317	0.7804	0.3212	0.1255
0.7	0.6488	0.7891	0.7623	0.6921	0.4911	0.1589
0.6	0.8638	0.9174	0.8160	0.7339	0.5431	0.1499
0.5	0.9838	1.0053	0.7864	0.7406	0.5577	0.2261

Table 10: The names of the algorithms.

English abbreviation	Full name
COS&TPRP-CF	Collaborative filtering algorithm based on cosine and traditional item rating prediction
U&TPRP-CF	Collaborative filtering algorithm based on user and traditional item rating prediction
U&URWFRP-CF	Collaborative filtering algorithm based on user and user scoring weight factor rating prediction
U&IPRP-CF	Collaborative filtering algorithm based on user and improved item rating prediction (our algorithm)

five algorithms are almost similar. But the error gap in TripAdvisor_v1 is obvious, and the multidimensional algorithms produce a high number of errors. MAE and RMSE calculate the difference between the predict rating and the user's true rating, which aims to measure the difference between the recommended result and the user's true preference. The smaller the MAE and the RMSE, the more users like the recommended items. Association rule mining (ARM) improves the precision and recall of the recommendation, which means that it improves click volume and purchase amount. At the same time, it also makes it difficult for the recommender system to predict the rating, and the probability of the system recommending items that the user does not like is increased. From the experiment results, it can reduce MAE and RMSE with improved item-rating prediction (IIP) method.

In general, the fusion algorithm Multi-CF-AR-IIP has better recommendation performance than the others. It

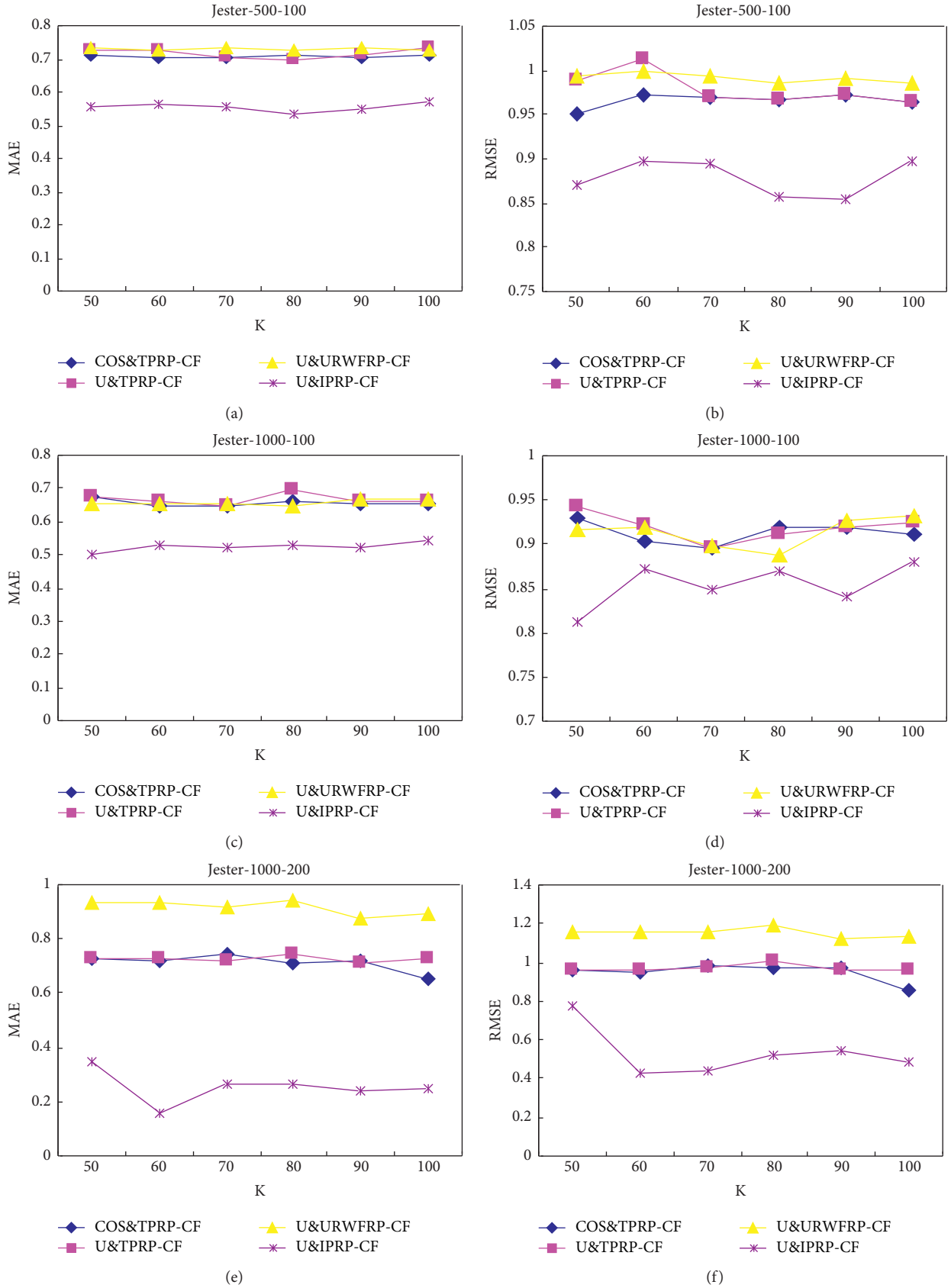

Figure 2: Continued.

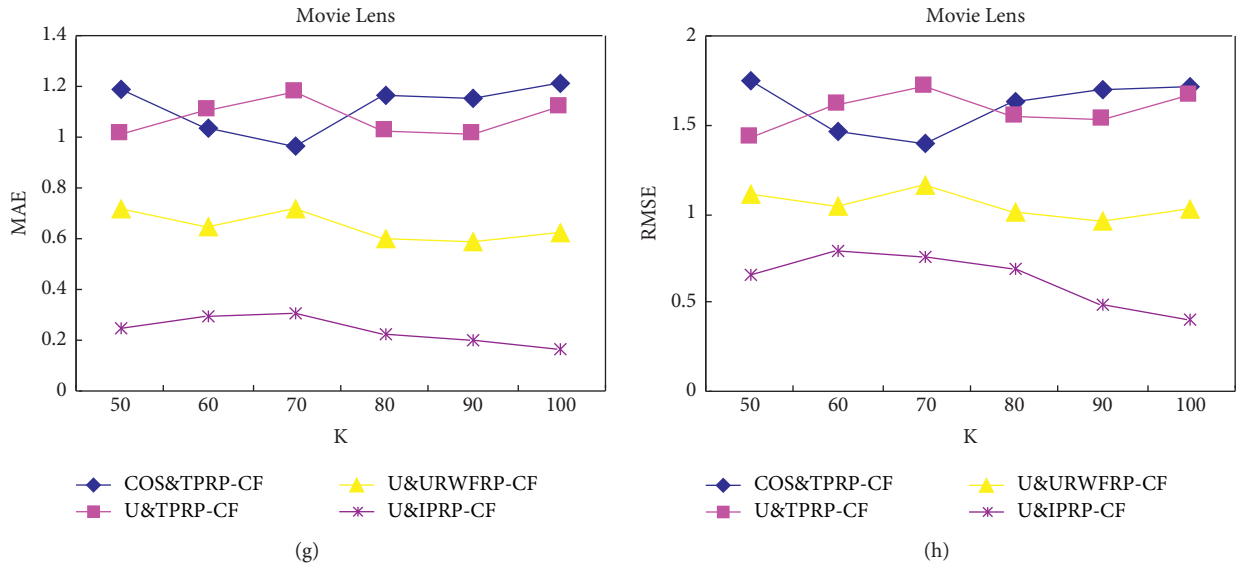

Figure 2: Comparison of the performance results for the different datasets. (a) Comparison of MAE for Jester-500-100. (b) Comparison of RMSE for Jester-500-100. (c) Comparison of MAE for Jester-1000-100. (d) Comparison of RMSE for Jester-1000-100. (e) Comparison of MAE for Jester-1000-200. (f) Comparison of RMSE for Jester-1000-200. (g) Comparison of MAE for MovieLens. (h) Comparison of RMSE for MovieLens.

Table 11: Information on the related context dimensions for the datasets.

DePaulMovie	Number	Users	Items	Ratings	Sparsity
		97	79	5043	0.3419
	Context dimension	Time	Location		Companion
		Weekday, weekend	Cinema, home		Alone, family, partner
TripAdvisor_v1	Number	Users	Items	Ratings	Sparsity
		1129	48	4669	0.9138
	Context dimension	USER_TIMEZONE	HOTEL_TIMEZONE		Trip type
		Eastern, central, pacific, mountain, HI, AK	Eastern, central, pacific, mountain		1, 2, 3, 4, 5

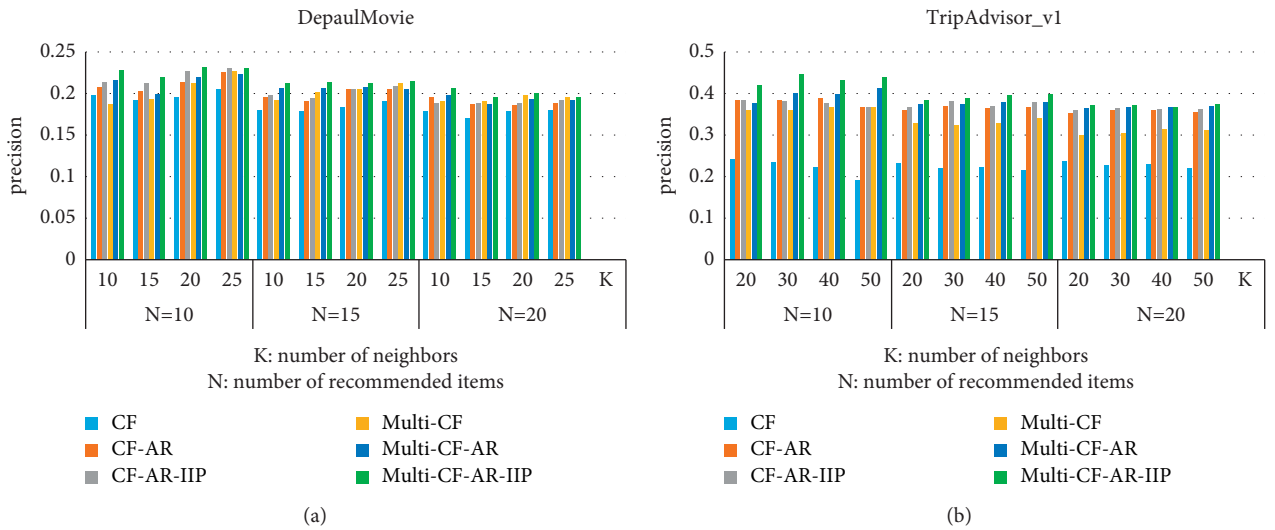

Figure 3: Continued.

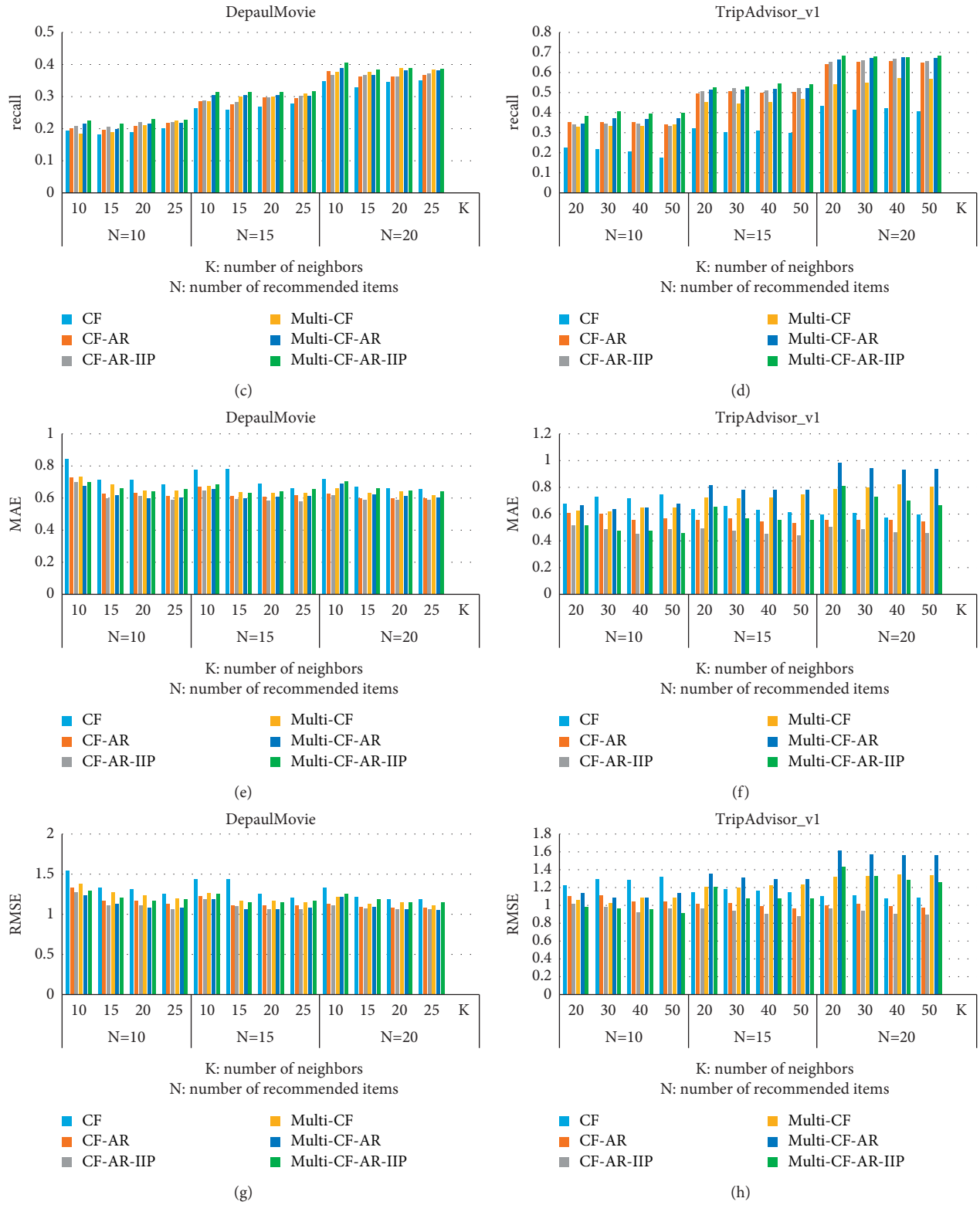

Figure 3: Performance comparison of algorithms based on different datasets. (a) Comparison of precision of algorithms based on DePaulMovie. (b) Comparison of precision of algorithms based on TripAdvisor_v1. (c) Comparison of recall of algorithms based on DePaulMovie. (d) Comparison of recall of algorithms based on TripAdvisor_v1. (e) Comparison of MAE of algorithms based on DePaulMovie. (f) Comparison of MAE of algorithms based on TripAdvisor_v1. (g) Comparison of RMSE of algorithms based on DePaulMovie. (h) Comparison of RMSE of algorithms based n TripAdvisor_v1.

recommends more diversified items to users by using multidimension context and AR and recommends items that users may prefer by using IIP.

4. Conclusion

In recent years, recommendation systems have been widely used in various fields. The accuracy and applicability of the recommendation system is very important. In this paper, we proposed a novel recommendation algorithm based on improved collaborative filtering with multidimensional context and association rules. Firstly, an improved cross-iterative bi-clustering based user scoring prediction method is proposed. Then, the multidimensional context-aware method is introduced into the traditional user collaborative filtering algorithm by using context-aware related information. In this method, a multidimensional context is used to filter the original data, the excess data is filtered to adjust the recommendation results, and the context data is integrated into the similarity calculation process of the user and the product to obtain more accurate recommendation results. In addition, in order to better compensate for the impact of data sparseness and increase the user's satisfaction, association rules can be used to find similar preferences between users with low similarity. By mining the context and the relevance of the user's selected items, we can find popular items with a high degree of contextual relevance to complement the algorithm's novelty and reliability. The algorithm proposed in this paper can enhance the user's experience on the recommendation platform and strengthen the connection between context and recommendation results. The algorithms we propose provide recommendations for users in a multidimensional context environment, which not only complements the omission of the collaborative filtering algorithm, but also improves the accuracy and efficiency of the recommendation results.

In the future, we will study high-dimensional clustering algorithms, which will help solve the problem of data sparsity and determine the decision-making of social groups. To establish a more personalized recommendation system, we must develop effective recommendation methods from multiple perspectives. Another new research direction is how to use recursive neural networks to provide personalized advice.

Data Availability

Data sharing is not applicable to this article as no datasets were generated or analysed during the current study.

Conflicts of Interest

The authors declare no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Acknowledgments

This work was supported by Fund Item of the China Scholarship Council (CSC) and Key Laboratory Project of Sichuan University (QXXCSYS201705).

References

[1] A. K. Sahu and P. Dwivedi, "User profile as a bridge in cross-domain recommender systems for sparsity reduction," *Applied Intelligence*, vol. 49, no. 7, pp. 2461–2481, 2019.

[2] L. Xu, C. Jiang, Y. Chen, Y. Ren, and K. J. R. Liu, "User participation in collaborative filtering-based recommendation systems: a game theoretic approach," *IEEE Transactions on Cybernetics*, vol. 49, no. 4, pp. 1339–1352, 2018.

[3] T. Xiao and H. Shen, "Neural variational matrix factorization for collaborative filtering in recommendation systems," *Applied Intelligence*, vol. 49, no. 4, pp. 3558–3569, 2019.

[4] B. Loepp, T. Donkers, T. Kleemann, and J. . Ziegler, "Interactive recommending with tag-enhanced matrix factorization (TagMF)," *International Journal of Human-Computer Studies*, vol. 121, pp. 21–41, 2018.

[5] H. Luo, M. Li, S. Wang, Q. Liu, Y. Li, and J. Wang, "Computational drug repositioning using low-rank matrix approximation and randomized algorithms," *Bioinformatics*, vol. 34, no. 11, pp. 1904–1912, 2018.

[6] W. Wang, J. Chen, J. Wang, J. Chen, J. Liu, and Z. Gong, "Trust-enhanced collaborative filtering for personalized point of interests recommendation," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 9, pp. 6124–6132, 2020.

[7] N. V. Dat, P. V. Toan, and T. M. Thanh, "Solving distribution problems in content-based recommendation system with Gaussian mixture model," *Applied Intelligence*, vol. 5, pp. 1–13, 2021.

[8] W. Zhang, X. Zhang, and D. Chen, "Causal neural fuzzy inference modelling of missing data in implicit recommendation system," *Knowledge-Based Systems*, vol. 222, no. 10, pp. 66–78, 2021.

[9] A. A. Amer, H. I. Abdalla, and L. Nguyen, "Enhancing recommendation systems performance using highly-effective similarity measures," *Knowledge-Based Systems*, vol. 217, Article ID 106842, 2021.

[10] M. Liu, W. Pan, M. Liu, Y. Chen, X. Peng, and Z. Ming, "Mixed similarity learning for recommendation with implicit feedback," *Knowledge-Based Systems*, vol. 119, no. 1, pp. 178–185, 2017.

[11] Z. Duan, W. Xu, Y. Chen, and L. Ding, "Etbrec: a novel recommendation algorithm combining the double influence of trust relationship and expert users," *Applied Intelligence*, vol. 2021, no. 3, 2021.

[12] X. Kong, F. Xia, J. Wang, A. Rahim, and S. k. Das, "Time-location-relationship combined service recommendation based on taxi trajectory data," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 3, pp. 1202–1212, 2017.

[13] W. X. Zhao, S. Li, Y. He, L. Wang, J. Wen, and X. Li, "Exploring demographic information in social media for product recommendation," *Knowledge and Information Systems*, vol. 49, no. 1, pp. 1–25, 2015.

[14] Y. Wang and X. Li, "Study on improved clustering collaborative filtering algorithm based on demography," *Computer Science*, vol. 44, no. 3, pp. 63–69, 2017.

[15] S. Mandal and A. Maiti, "Deep collaborative filtering with social promoter score-based user-item interaction: a new perspective in recommendation," *Applied Intelligence*, vol. 51, pp. 1–26, 2021.

[16] Y. Pan, Y. Huo, J. Tang, Y. Zeng, and B. Chen, "Exploiting relational tag expansion for dynamic user profile in a tag-aware ranking recommender system," *Information Sciences*, vol. 545, no. 6, pp. 448–464, 2021.

[17] J. Gou, J. Guo, L. Zhang, and C. Wang, "Collaborative filtering recommendation system based on trust-aware and domain experts," *Intelligent Data Analysis*, vol. 23, no. S1, pp. 133–151, 2019.

[18] X. Y. Xu, L. H. Ren, and Y. S. Ding, "An improved D-S evidence theory based on genetic algorithm to VIP intelligent recognition and recommendation system," *Applied Mechanics and Materials*, vol. 347-350, pp. 2442–2446, 2013.

[19] J. Xiao, M. Luo, J.-M. Chen, and J.-J. Li, "An item based collaborative filtering system combined with genetic algorithms using rating behavior," *Lecture Notes in Computer Science*, vol. 9227, pp. 453–460, 2015.

[20] Y. Lin, P. Ren, Z. Chen, Z. Ren, J. Ma, and M. De Rijke, "Explainable outfit recommendation with joint outfit matching and comment generation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 8, pp. 1502–1516, 2018.

[21] A. Mcy, B. Ik, and B. Ksl, "Temporal context-aware task recommendation in crowdsourcing systems," *Knowledge-Based Systems*, vol. 219, Article ID 106770, 2021.

[22] Z. E. Yebdri, S. M. Benslimane, F. Lahfa, M. Barhamgi, and D. Benslimane, "Context-aware recommender system using trust network," *Computing*, vol. 2021, no. 103, pp. 1919–1937, 2021.

[23] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," *Uncertainty in Artificial Intelligence*, vol. 98, no. 7, pp. 43–52, 2013.

[24] B. Hui, L. Zhang, X. Zhou, X. Wen, and Y. Nian, "Personalized recommendation system based on knowledge embedding and historical behavior," *Applied Intelligence*, vol. 2021, no. 7, 2021.

[25] Y. Zheng, B. Mobasher, and R. Burke, "CARSKit: a java-based context-aware recommendation engine," in *Proceedings of the 15th IEEE conference on Data Mining Workshops*, pp. 1668–1672, IEEE, Atlantic City, NJ, USA, Nov-2015.

[26] Y. Zheng, R. Burke, and B. Mobasher, "Differential context relaxation for context-aware travel recommendation," in *Proceedings of the 13th International Conference on Electronic Commerce and Web Technologies (EC-WEB 2012)*, pp. 88–99, Springer, Chicago City, IL, USA, Sep-2012.