LSH-based Collaborative Recommendation Method with Privacy-Preservation

Jiangmin Xu[1], Xuansong Li[1], Hao Wang[2], Hong-Ning Dai[3], Shunmei Meng[1]*
[1] Department of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China
[2]Department of Computer Science, Norwegian University of Science and Technology, 2815 Gjvik, Norway
[3]Faculty of Information Technology, Macau University of Science and Technology, Macau, China
E-mail: xwhite169@163.com, lixs@njust.edu.cn, hawa@ntnu.no, hndai@ieee.org, mengshunmei@njust.edu.cn*

***Abstract*—With the rapid development of cloud computing technology, massive services and online information cause information overload. Collaborative Filtering (CF) is one of the most successful and widely used technologies in personalized recommendation system to deal with information overload. However, traditional CF recommendation algorithms go through high time cost and poor real-time performance when dealing with the large-scale behavior data. Moreover, most collaborative recommendation methods mainly focus on improving recommendation accuracy, while ignore privacy preservation. In addition, the recommendation results of traditional CF recommendation algorithms are often too single, which could not meet user's diverse requirements. To solve these problems, this paper proposes a privacy-aware collaborative recommendation algorithm based on local sensitive hash (LSH) and factorization techniques. First, LSH is adopted to determine nearest neighbor set of the target users, where a neighbor matrix for the target user can be generated. The matrix factorization technique is applied in the neighbor matrix to predict the missing ratings. Then the nearest neighbors can be determined based on the predicted ratings. Finally, predictions for the target user are made based on the neighborhood-based CF recommendation model and diversified recommendations are made for the target user. Experimental results show that the proposed algorithm can effectively improve the efficiency of recommendation on the premise of protecting the privacy of users.**

Keywords- Cloud computing; collaborative recommendation; local sensitive hash; matrix factorization

I. INTRODUCTION

Rapid development of Internet technologies and cloud techniques has brought great convenience to our daily life [1-2]. However, massive cloud services in the cloud platform also cause information overload. With increased items and online information brought by cloud service, it becomes more and more difficult for users to find ideal results. Traditional information retrieval algorithms may not meet the growing needs of users to obtain personalized information. In this case, personalized recommendation system comes into being [3-5].

At present, personalized recommendation technology can be mainly divided into three categories: content-based recommendation [6], collaborative filtering (CF) recommendation [7] and hybrid recommendation[8]. However, there are still many fundamental tasks to be addressed, such as real-time recommendation, the balance between recommendation accuracy and the algorithm complexity. Neighborhood model is one of the most successful CF recommendation techniques, including user-based collaborative filtering and item-based collaborative filtering. The recommendation efficiency of neighborhood-based CF algorithms mainly depends on the accuracy of similarity measurement [9].

With the proliferation of users and online information in cloud environment, the scale of recommendation system is gradually expanding. Traditional CF algorithms are not effective to deal with large-scale, high-dimensional and sparse user rating data [10]. As a result, the adaptability and real-time performance of recommendation algorithm are seriously affected, which further affects the recommendation results. To solve these problems, many effective methods have been proposed [11-12]. In reference [13], a new forecasting and computing model based on the matrix factorization model is proposed, but this method still directly uses the traditional similarity measurement to calculate the similarity between users, without considering the impact of user or item category information on recommendation. In researches [14-16], the authors apply the clustering techniques into recommendation models to reduce the time of neighbor searching and improve prediction accuracy. However, clustering analysis faces with problems, such as multi-dimensional clustering categories, and is difficult to control the measurement standards. The factorization technique has been proved to be an effective tool to deal with the sparsity problem. However, when dealing with large-scale or high-dimensional data matrix, the traditional factorization techniques are not so effective and the recommendation efficiency is not ideal. Moreover, when dealing with large-scale rating data, the traditional CF algorithms cause large amount of offline calculation with weak scalability, which will seriously affect the recommendation efficiency.

In addition, the historical rating data may contain user's privacy information, which may be illegally used or even resold. However, most existing CF recommendation systems seldom consider privacy preservation, which greatly increases the risk of user privacy disclosure and reduces the enthusiasm of users [18-19]. Moreover, most existing recommendation systems tend to pay more attention to the accuracy of the recommendation results. This single recommendation standard may lead to the redundancy of the recommendation results, so that users will have aesthetic fatigue and feel disappointed.

In view of the above challenges, this paper proposes an improved collaborative recommendation algorithm based on LSH and matrix factorization technique. This method can effectively adapt to high-dimensional and sparse user rating

2159-6190/20/$31.00 ©2020 IEEE
DOI 10.1109/CLOUD49709.2020.00085

data. The LSH technique is effective to speed up the neighbor searching. In addition, by the LSH-based mapping technique, the hash values, not the original rating data, are utilized for neighbor searching, thus user's sensitive information is preserved. The matrix factorization technique is applied to predict missing rating data, and then predictions are made based on the improved neighborhood-based CF model. Finally, diversified recommendation results are presented. Compared with the traditional recommendation algorithms, the experimental results show that the algorithm proposed in this paper gets greater advantages.

The remainder of the paper is organized as follows: Section II provides some preliminary knowledge. Section III proposes an improved collaborative filtering recommendation algorithm based on LSH and matrix factorization techniques. Section IV presents the experimental results to validate the effectiveness of this method. Some related researches are reviewed in Section V. Finally, Section VI concludes the paper and gives an outlook on possible directions of our work.

II. PRELIMINARY KNOWLEDGE

A. Use- based CF recommendation algorithm

The basic idea of user-based CF recommendation algorithm is to find a set of neighbor users with high similarity to the interest of the target user according to the historical behavior information of the target user, and then predict the corresponding rating of the target user according to the ratings of the neighbor users for items, and select top-*N* items which have highest ratings to recommend to the target user. Suppose that user A likes item a, item b and item d, user B likes item c, and user C likes item a and item d. From these user historical preference data, it can be found that both user A and user C like item a and item d, and their preferences are similar, so it can also be assumed that user C will like item b, so item b is recommended to user C. Figure 1 shows the principle of this algorithm.

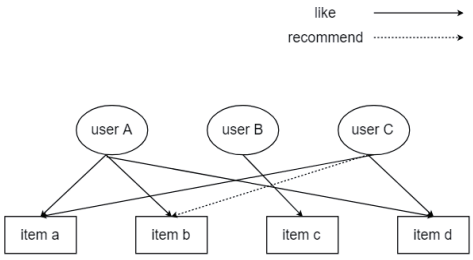

Figure 1 User based CF recommendation algorithm principle

Firstly, the rating matrix R^{m*n} of users is established according to the data set, where *m* refers to the number of users in the matrix and *n* refers to the number of items. The main step of user-based CF recommendation algorithm is to find the nearest neighbors (the users that have the highest similarities with the target user) of the target user. The target user's rating for unrated items is predicted by the nearest neighbors' rating on items. The similarity calculation methods in CF recommender systems mainly include cosine similarity, improved cosine similarity, Pearson correlation coefficient and so on. Herlocker et al. [20] carry out experiment showing that Pearson correlation coefficient is most suitable for user-based CF recommendation algorithm, which is as follows:

$$sim(u,v)=\frac{\sum_{i\in I}(r_{ui}-\overline{r_u})(r_{vi}-\overline{r_v})}{\sqrt{\sum_{i\in I}(r_{ui}-\overline{r_u})^2}\sqrt{\sum_{i\in I}(r_{vi}-\overline{r_v})^2}} \quad (1)$$

where, I represents the set of items rated by user u and user v together; r_{ui} represents the rating of user u on item i; rep r_{vi} resents the rating of user v on item i. $\overline{r_u}$ and $\overline{r_v}$ represent the average value of user u and user v for all rated items.

B. Local sensitive hash

The local sensitive hash technology [21] is suitable for fast approximate query under huge amount of data, and the accuracy of query results is basically the same as that of "brute force" query. It is widely used in video, image and other fields with rich information.

Local sensitive hash means that through hash mapping, not only the original real QoS data can be hidden, but also it has a good "similarity retention" before and after mapping That is, the two points that are originally similar have a high probability of being similar after hash, and vice versa. As shown in Fig. 2, after LSH mapping, points A and B that are originally similar will be mapped to the same bucket in the hash table , points C and D that are not originally similar will be mapped to different buckets in the hash table.

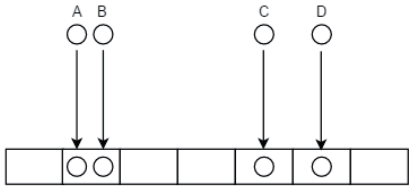

Figure 2 Local sensitive hash

Specifically, according to the principle of local sensitive hash, the original adjacent data points are still adjacent after hash mapping, that is, they have the same bucket number. Once the original data has been hashed, the resulting hash value is stored in the hash table. When the original data is converted to different bucket numbers of hash table, the data points with the same bucket number are most likely to be originally adjacent. At the same time, there may be data points that are not adjacent to each other in the original data falling into the same bucket. Therefore, the hash function needs to meet the following conditions:

$$if\ \ d(x,y)\le d_1, then\ P(h(x)=h(y))\ge P_1 \quad (2)$$

$$if\ \ d(x,y)\ge d_2, then\ P(h(x)=h(y))\le P_2 \quad (3)$$

where x and y represent the original data points, and $d(x, y)$ represents the distance between the original data points x and y. P_1, P_2 represent probability threshold, d_1, d_2 represent distance threshold, $h(x)$, $h(y)$ represent hash function, and hash map data points. If $d(x, y) \le d_1$, then P_1 is the minimum probability of $h(x) = h(y)$. If $d(x, y) \ge d_2$, then the probability of $h(x) = h(y)$ is at most P_2, and $P_1 \le P_2$. Only when the hash function satisfies the above formulas can the original data points be transformed effectively. After getting the hash function, a hash list can be made up. After multiple hash transformations, the original data points are finally converted to 0 or 1 in the binary bits and put into the bucket.

The process of using LSH to index massive data (hash table) and searching user's nearest neighbor through the index previously established is as follows:

1) Build user index offline

- Select the hash function that meets the requirements.
- Map the raw data hash into different buckets.

2) Search online

- Map the query data through using the hash function to get the corresponding bucket number.
- Find the nearest neighbor in the bucket.

Based on the characteristics of LSH itself, the search of the nearest neighbor of the target user only needs to search in the bucket instead of traversing the entire data set. Because the number of data points in the bucket is far smaller than the number of original data, the search efficiency is greatly improved. In addition, after hash mapping, the reduced dimension data, not the original rating data, is utilized for the next processing, which will not disclose the sensitive information of users. In other words, the recommender systems only know the hash values, not the original rating data, thus the personal privacy information of users will be protected. In the case of dealing with massive high-dimensional data and non-disclosure of users' privacy information, local sensitive hash method has great advantages.

C. Matrix Factorization

The user's rating matrix is often a very sparse matrix. In order to fill in the missing values of the matrix, the matrix factorization technology is used.

(1) BasicSVD

BasicSVD is the simplest matrix factorization method. It decomposes the rating matrix into two low-order matrices, representing the implicit characteristics of users and items, as follows:

$$R^{'} = p_{uk} \cdot q_{ik}^{T} \tag{4}$$

where u is the number of users, i is the number of items, and k is the number of implied features. By minimizing the sum of squared errors (SSE), the implicit feature matrix p and q are obtained, and then the rating is predicted by the implicit feature matrix:

$$SEE = \sum_{u,i} (R - R^{'})^2 = \sum_{u,i} (R - p_{uk} \cdot q_{ik}^{T})^2 \tag{5}$$

The optimization objective is to minimize SSE, and the parameters are matrix p, q:

$$\min_{p,q} SEE \tag{6}$$

(2) FunkSVD

FunkSVD adds regularization items on the basis of basicSVD to prevent overfitting. The optimization objective is to minimize the sum of SSE and regularization terms, and the parameters are matrix p, q.

$$\min_{p,q} SEE + \lambda(\|p_u\|^2 + \|q_i\|^2) \tag{7}$$

The matrix factorization algorithm usually uses the stochastic gradient descent method to train the model. It needs to calculate the partial derivative of the loss function for each parameter to get the parameter updating formula and update the parameter iteratively until the algorithm converges.

III. Improved Collaborative Recommendation Method Based on LSH and Factorization Techniques

Based on the above, this paper proposes an improved collaborative filtering recommendation algorithm based on LSH and factorization techniques. Our method consists of four phases. Firstly, we select the appropriate hash function family to build the user index offline, and the target user is hashed online to get the corresponding bucket number. Then in each bucket, the rating data of all users in the bucket are formulated as a user-item rating matrix, which is decomposed to predict the missing rating data. Ratings are predicted based on the nearest neighbors . Finally, diversified recommendation results are presented.

A. Determine the hash buckets based on LSH

(1) Build user's index offline

In the traditional collaborative filtering method, PCC is usually used to measure the similarity between users. In the definition of LSH, hash function and hash function family are used to determine the similarity between users. Through the definition of the hash functions in formula (2) and formula (3), the hash function $h(u)$ is selected and the hash function family H is established. Since PCC is often used as a method to calculate similarity or measure distance in recommendation system, this paper uses LSH function corresponding to PCC to build index. The history rating vector of users can be expressed as an n-dimensional vector $\vec{u} = (u_1, u_2, ..., u_n)$, where u_i is the user's rating for the i-th item. If the user has not rated the i-th item, the u_i 's value is set as 0. In this case, a new random n-dimensional vector $\vec{v} = (v_1, v_2, ..., v_n)$ is created, where the value of v_i is randomly generated .It is a random number within [-1, 1]. Then we set the hash function family $H = (h_1, h_2, ..., h_m)$, where h is:

$$h_i = 1, \vec{u} \cdot \vec{v} > 0 \text{ else } 0 \tag{8}$$

Through a hash function $h_i(u)$ and a random vector $\vec{v}$ in a hash function family H, the n-dimensional user-item history rating vector will be mapped to a binary value as 0 or 1. Apply the m hash functions in the hash function family to the user-item history rating vector in the above way, we can get an m-dimensional binary vector, that is, we map the n-dimensional user-item history rating vector to an m-dimensional binary vector (or the integer value represented by it) through the above local sensitive hash operation, and the original user-item history rating vectors which have the same mapping result are likely to be similar. Considering the local sensitive hash is a probability-based model, we usually create multiple hash tables ,but each hash table is implemented in the same way, they only use different random vectors.

(2) Find the hash bucket of the target user

First of all, through the hash function family $H=(h_1,h_2,...,h_m)$ selected in step (1) and the rating vector of the target user, we can calculate the index of the target user online, that is, the bucket number, which is an m-dimensional binary vector. Then, based on the hash table established offline in the first step, the users with the same bucket number of the target user are regarded as the similar neighbors of the target user and are stored in the neighbor set. Through a hash table, we can find a neighbor set of the target user. Next, for all hash tables, we can find the corresponding neighbor sets according to the above steps. Finally, we take the union of these neighbor sets as the final nearest neighbor set of the target user. And a neighbor matrix for the target user can be generated based on the nearest neighbor set.

B. Predict missing rating data based on MF

When the collaborative filtering algorithm calculates the user similarity, in most cases, it will exclude users with high similarity but having ratings missed and the user-item rating matrix is usually spare. Therefore, we perform matrix factorization in the neighbor matrix to predict the missing rating first.

The user's $m\times n$-dimensional rating matrix R is decomposed into two low-dimensional matrix multiplication:

$$R=P^TQ \tag{9}$$

where $P\in R^{f\times m}$ and $P\in R^{f\times n}$ are two reduced dimensional matrices. Then, the predicted value of user u's rating for item i, $r_{ui}^{'}=R(u,i)$ can be calculated by the following formula:

$$r_{ui}^{'}=\sum_f p_{uf}q_{if} \tag{10}$$

The loss function is:

$$C(p,q)=\sum_{(u,i)\in Train}(r_{ui}-r_{ui}^{'})^2=\sum_{(u,i)\in Train}(r_{ui}-\sum_{f=1}^{F}p_{uf}q_{if})^2 \tag{11}$$

Direct optimization of the above loss function may lead to overfitting of learning. Therefore, it is necessary to add the item $\lambda(\|p_u\|^2+\|q_i\|^2)$ to prevent over fitting, where λ is the regularization parameter, so as to obtain:

$$C(p,q)=\sum_{(u,i)\in Train}(r_{ui}-\sum_{f=1}^{F}p_{uf}q_{if})^2+\lambda(\|p_u\|^2+\|q_i\|^2) \tag{12}$$

To minimize the above loss function, we can use the stochastic gradient descent method. This method finds the direction of the fastest descent by calculating the partial derivative of the model parameters, and then optimizes the model parameters by iterative method. In the loss function defined above, there are two sets of model parameters, p_{uf} and q_{if}. Firstly, the gradient descent method needs to calculate the partial derivatives of them respectively, which can be obtained as follows:

$$\begin{aligned}\frac{\partial C}{\partial p_{uf}}&=-2q_{ik}\cdot(r_{ui}-r_{ui}^{'})+2\lambda p_{uk}\\ \frac{\partial C}{\partial q_{if}}&=-2p_{uk}\cdot(r_{ui}-r_{ui}^{'})+2\lambda q_{ik}\end{aligned} \tag{13}$$

Then, according to the stochastic gradient descent method, it is necessary to update the model parameters along the direction of the fastest descent, so the following recurrence formula can be obtained:

$$\begin{aligned}p_{uf}&=p_{uf}+\alpha(q_{ik}\cdot(r_{ui}-r_{ui}^{'})-\lambda p_{uk})\\ q_{if}&=q_{if}+\alpha(p_{uk}\cdot(r_{ui}-r_{ui}^{'})-\lambda q_{ik})\end{aligned} \tag{14}$$

where α is the learning rate, and its value needs to be obtained through repeated experiments.

C. Rating Prediction

(1) Find the k nearest neighbor of the target user

For all users in the nearest neighbor target group, the similarity between them and target user needs to be calculated. Pearson correlation coefficient is used for similarity calculation, and the calculation formula is as follows:

$$sim(u,v)=\frac{\sum_{i\in I}(r_{ui}-\overline{r_u})(r_{vi}-\overline{r_v})}{\sqrt{\sum_{i\in I}(r_{ui}-\overline{r_u})^2}\sqrt{\sum_{i\in I}(r_{vi}-\overline{r_v})^2}} \tag{15}$$

where I represents the set of items rated by user u and user v together; r_{ui} represents the rating of user u on item i; r_{vi} represents the rating of user v on item i. $\overline{r_u}$ and $\overline{r_v}$ represent the average value of user u and user v for all rated items. Then we get the user sets $S(u,k)$ with k users of highest similarity.

(2) Rating prediction and recommendation

In the user based collaborative filtering recommendation algorithm, to predict a user's rating of an item, we need to refer to the ratings of the item rated by users who have similar interests with the target user, that is:

$$r_{ui}^{'} = \overline{r_u} + \frac{\sum_{v \in S(u,k) \cap N(i)} w_{uv}(r_{vi} - \overline{r_v})}{\sum_{v \in S(u,k) \cap N(i)} |w_{uv}|} \quad (16)$$

Among them, $S(u,k)$ is the set of k users who are most similar to user u, $N(i)$ is the set of users who have rated item i, r_{vi} is user v's rating of item i, and $\overline{r_v}$ is the average value of user v's rating of all items that he has rated. Based on the predicted rating value of the item which has not been rated by the target user, the predicted value is sorted in descending order. Then the Top-k items are recommended to the target user.

D. Diversity of recommended results

In recommendation systems, items often have various labels. For example, in a movie recommendation system, a movie usually has multiple labels, such as animation, children's, comedy, adventure, etc.If we do not consider the multiple types of movie tags, and only consider the perspective of accuracy, the recommendation list generated by recommender system may have a large number of movies with the same tag type, while the single and repeated movies may reduce the satisfaction of users.

The basic idea of the diversification method of recommendation results is to use the recommendation list generated to further diversify the movies in the recommendation list according to the movie tags. In the above step, the Top-k recommendation list has been generated for users. Based on this, according to the movie tags in the recommendation list, the result diversification method is applied to reduce the single recommended items and improve the diversity of recommended items. The details are described as follows:

Set the recommended list as $\{I_1, I_2, ..., I_K\}$, the labels diversity of items I_i and I_j are defined as follows:

$$P_{ij} = \frac{|T_i \cap T_j|}{|T_i \cup T_j|} \quad (17)$$

where T_i is the label set of item i. The smaller the value of P_{ij}, the less overlapped tags in the two item tag sets. In other words, the smaller the value of P_{ij} is, the more diverse the two items are. If $P_{ij} = 0$, it means that the labels in the two item label sets T_i and T_j do not coincide. If $P_{ij} = 1$, it means that the labels in the movie label set T_i and T_j are the same. For the items in the recommendation list, the label diversity P_{ij} of any two items is calculated. Then, items are sorted according to the value of P_{ij}. We filter items with the same label, and then diversify the entire recommendation list. Finally, the filtered list of items, i.e., the regenerated recommendation list, is recommended to the target users.

IV. Experiment

In this section, we analyze and verify the performance of the proposed algorithm through experiments.

A. Experimental dataset

In order to evaluate the performance of the algorithm, the Movielens data set is used in the experiment. Movielens is a rating-based movie recommendation system, created by the GroupLens research group of the University of Minnesota in the United States (http://www.grouplens.org/node/73). It includes 100KB, 1MB, 10MB and other data sets of various sizes, which are dedicated to research recommendation technology. This paper uses Movielens 1M data set to verify the performance of the algorithm. The data set contains one million ratings of 3952 movies by 6040 users, in which each user has rated at least 20 movies, with a rating range of 1-5. The higher the rating of a movie is, the more the user likes the movie.

This paper uses cross validation to determine the validity of the model, and analyzes the results. The original dataset is randomly divided into two complementary subsets according to the scale of 4:1, 80% of which is the training set, and the remaining 20% is the test set. Firstly, the training set is analyzed, and then the impact on the test set is verified. In order to reduce the variability, this paper uses different partition to carry out multiple cross validation, and averages the results of each validation. In this paper, the experiment carries out a five-fold cross validation.

In order to evaluate the performance of the recommended algorithm, the algorithm proposed in this paper is compared with the following methods: UPCC [22], IPCC [23], P-UIPCC [24].

B. Evaluation metrics

In order to measure the performance of the method proposed in this paper, MAE (mean absolute error), Time Cost, Precision and Recall are used as evaluation indexes.

The MAE metric shows the average error between the prediction and the actual value. The lower the value is, the better the recommendation effect of the recommendation system is. MAE is defined as follows:

$$MAE = \frac{\sum_{i,j \in T} |r_{i,j} - r_{i,j}^{'}|}{|T|}$$

where $r_{i,j}$ represents the actual rating of user i for item j, $r_{i,j}^{'}$ represents the predicted rating of user i for item j, T represents the test set, and $|T|$ is the number of ratings included in the test set.

Time cost means the time cost of generating recommendation list. The smaller the time cost value is, the shorter the time to generate the recommendation list is, that is, the higher the recommendation efficiency is.

Precision indicates the proportion of items that users actually like in the items recommended by the recommendation system for users. The larger the proportion is, the higher the recommendation accuracy of the algorithm is. Precision is defined as:

$$Precision = \frac{\sum_{u \in U} |R(u) \cap T(u)|}{\sum_{u \in U} |R(u)|}$$

Among them, $R(u)$ refers to the list of recommendations made to users according to their behaviors on the training set, and $T(u)$ refers to the list of users' behaviors on the test set, that is, the items they like.

Recall refers to the proportion of items recommended by the recommendation system for users and liked by users in all users' favorite items, which is defined as:

$$Recall = \frac{\sum_{u \in U} |R(u) \cap T(u)|}{\sum_{u \in U} |T(u)|}$$

C. Experimental results and analysis

(1) Performance compared with other methods

In our experiment, to evaluate the prediction accuracy of our proposal, we compare our method (denoted as LSHRec) with other three methods. The comparison results are shown in Fig. 3-Fig. 6.

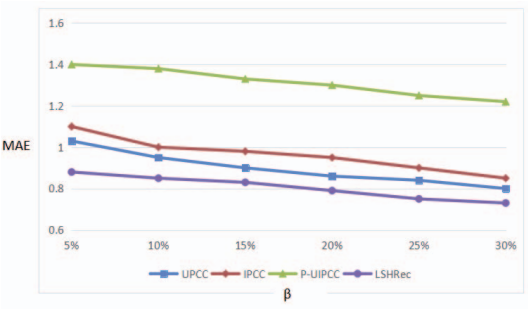

Figure 3 Performance comparison in MAE with the change of β

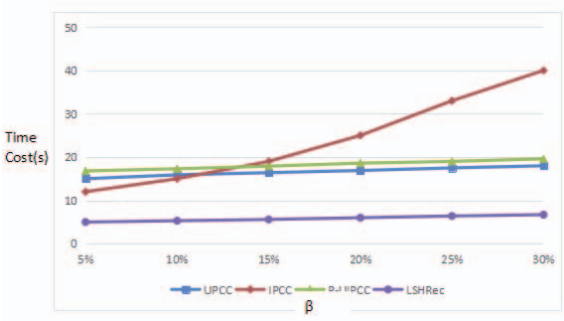

Figure 4 Performance comparison in Time Cost with the change of β

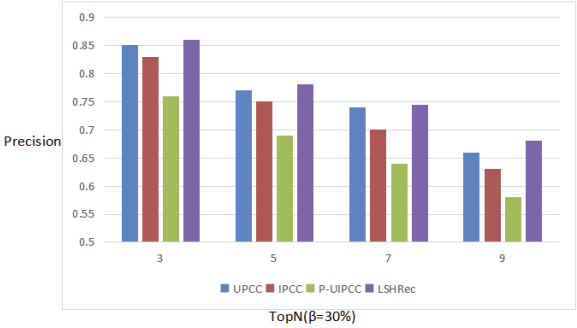

Figure 5 Comparison in Top-N prediction accuracy on Precision

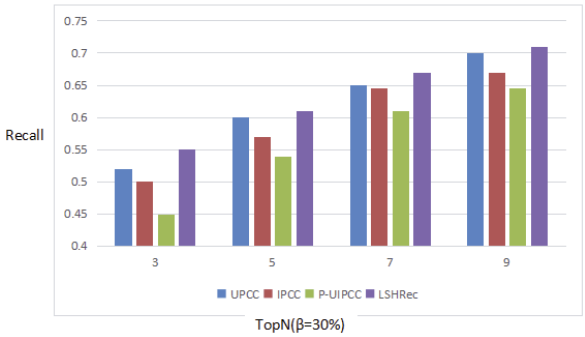

Figure 6 Comparison in Top-N prediction accuracy on Recall

Comparison in MAE and Time Cost: As shown in Fig. 3, β represents the percentage of data extracted from the data set (i.e. the density of scoring matrix), and the value range of β is 5% to 30%. In Fig. 3, the accuracy of the four methods increases with the increase of matrix density, because the overall accuracy of recommendations increases with the increase of the number of users and items. And we can see that our method performs better than P-UIPCC and has similar MAE with UPCC and IPCC. In Fig. 4, all of these methods cost more time with the increase of β. Among them, the time cost of IPCC increases rapidly with the increase of β while the time cost other three methods grows slowly. It also can be observed that our method consistently outperforms the other three methods. That is because that other methods needs to calculate all the similarity pairs, while our method could speed up neighbor searching only by using the LSH technique, which greatly improves the efficiency. Although LSH can speed up the search of neighbors and improve the efficiency of recommendation algorithm, it is a probability-based model, so that its accuracy may be affected. So we select the appropriate number of hash functions and hash tables and predict the missing rating data based on MF to ensure the accuracy of the algorithm.

Comparison in Precision and Recall: Fig. 5 and Fig. 6 depict the performance of Top-N (N = 3, 5, 7, 9) recommendations of all approaches, where β is set as 30%. Figure 5 shows the precision performance, and Fig. 6 shows the Recall performance. We can see that the precision of our method degrades with the increase of N, while the Recall upgrades. From Fig. 5 and Fig. 6, we can observe that our method, i.e., LSHRec, outperforms the other methods in both precision and recall. Although LSH may affect the accuracy to some degree since it's a probability-based model, however, we ensure its accuracy through predicting the missing rating data based on MF and the improved neighborhood-based CF model.

(2) Impact of parameter settings

In this section, we discuss the impact of parameter settings to prediction accuracy and time cost. Here, we mainly

analyze the impact of the number of hash tables, i.e., t and the number of hash functions in each hash table, i.e., m.

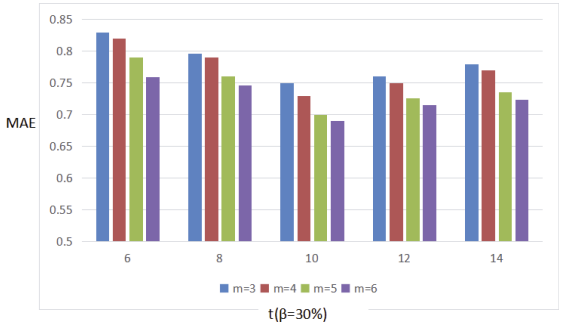

Figure 7 Parameters impact on prediction performance (MAE)

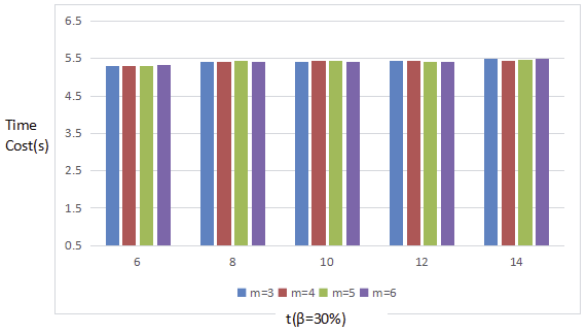

Figure 8 Parameters impact on Time Cost

In Fig. 7, t ranges from 6 to 14, m ranges from 3 to 6, and β is set as 30%. When the value of t is small, with the increase of the value of t, the accuracy of recommendation becomes higher. This is because more hash lists will increase the number of similar neighbors of the target user, so the accuracy of recommendation becomes better. When the value of t changes from 10 to 14, the accuracy of recommendation decreases. This is because in the algorithm design, the nearest neighbor set is the union of neighbor sets in different hash tables. With the increase of t, some users who are not similar may also enter the similar user set, and these noise data affect the accuracy of prediction, so in order to improve the accuracy of prediction, we should select an appropriate number of hash tables in practical application. And in this experiment, the most suitable value of t is 10.Besides, with the increase of the number of hash functions m, the prediction accuracy of our algorithm is improved. This is because in the algorithm design, with the increase of m, the design of hash function is stricter. Only users with high similarity can be mapped into the same hash bucket, thus forming the similar set of target user, which ensures the accuracy of recommendation results. Therefore, in order to improve the prediction accuracy, a large number of hash functions should be selected in practical application.

In Fig. 8, we can see that with the change of the number of hash tables t and the number of hash functions m, the time cost of recommendation basically almost remains the same. This is because the increase of the number of hash tables t and hash functions m will increase the time consumption to a certain extent, the increased time cost is negligible compared with the time saved by using LSH technology. Overall, the settings of t and m play an important role in improving the prediction accuracy of the recommendation algorithm, but have little effect on the time cost.

The above experimental results show that, our method can effectively improve the efficiency of recommendation algorithm and obtain satisfactory accuracy at the same time. The key to improve the accuracy of the algorithm is to select the appropriate value of t and m.

V. Related Work

This section reviews some related works. Li et al.[25] put forward an improved multi search local sensitive hash algorithm (mplsh) combined with item-CF recommendation algorithm. This method is to solve the problem that the computational efficiency of item-CF decreases sharply when processing high-dimensional item scoring data. In reference [26], a two-stage joint hash collaborative filtering algorithm is proposed. By mapping users and items to low-dimensional space while preserving users' preferences for projects, the process of recommendation is transformed into a project with a small Hamming distance between the low-dimensional space search and the target user. The authors in [27], aiming at the disadvantages of two-stage method, propose a kind of discrete optimization method. The quantization stage is integrated in the optimization process to avoid the coding loss in the optimization process as much as possible.

There also have been some recommendation algorithms associated with the LSH technique. Liu et al.[28] propose a k-nearest-neighbor collaborative filtering recommendation algorithm based on data dimensionality reduction and accurate Euclidean local sensitive hash. The algorithm focuses on too high scoring feature data dimension, slow searching speed of k-nearest-neighbor, and cold start of rating. Yan et al. [29] propose a hierarchical precise Euclidean local sensitive hash algorithm based on P-STABLE distribution to solve the problems of large scale, high sparsity of user rating data and poor real-time performance of direct similarity calculation in the classical collaborative filtering recommendation algorithm. Li et al. [30] propose a collaborative filtering recommendation algorithm based on the precise Euclidean local sensitive hash for the massive high-dimensional and sparse user rating data in the recommendation system, and study the impact on the recommendation quality caused by the large amount of computation and inaccurate results directly using the traditional similarity measurement method.

VI. Conclusion

In this paper, an improved collaborative recommendation algorithm based on LSH and matrix factorization technique is proposed. It aims to improve the efficiency of

recommendation system in the face of large-scale decision data with the consideration of privacy preservation. First, the users' hash buckets are determined based on the LSH technique and the privacy of users is protected. Then the missing rating data is predicted based on MF. Accordingly, target user's rating of the item is predicted based on the ratings of the item by users with similar interests. Finally, diversified recommendations are made for the target user to improve users' satisfaction with the recommendation results. Experiments are conducted based on the Movielens dataset. The experimental results show that compared with the traditional recommendation algorithm, the effectiveness of the recommendation algorithm proposed in this paper has been improved in dealing with large-scale decision data with the consideration of privacy-preservation. In our future work, we will focus on researches in improving both the efficiency and prediction accuracy of recommendation algorithms by integrating the deep learning techniques as the LSH technique adopted in this paper may affect the prediction accuracy.

ACKNOWLEDGEMENTS

This paper is partially supported by the National Natural Science Foundation of China under Grant No. 61702264, No. 61702263, No. 61761136003, the Fundamental Research Funds for the Central Universities under Grant No. 30918014108, the Postdoctoral Science Foundation of China under Grant No. 2019M651835.

REFERENCES

[1] Zhang J, Zhou Z, Li S, et al. Hybrid computation offloading for smart home automation in mobile cloud computing. Personal and Ubiquitous Computing, 2018, 22(1): 121-134.

[2] Qi L, Yu J, Zhou Z. An invocation cost optimization method for web services in cloud environment. Scientific Programming, 2017.

[3] Zhang S, Yao L, Sun A, et al. Deep learning based recommender system: A survey and new perspectives. ACM Computing Surveys (CSUR), 2019, 52(1): 1-38.

[4] Yan C, Cui X, Qi L, et al. Privacy-aware data publishing and integration for collaborative service recommendation. IEEE Access, 2018, 6: 43021-43028.

[5] Qi L, He Q, Chen F, et al. Finding all you need: web APIs recommendation in web of things through keywords search. IEEE Transactions on Computational Social Systems, 2019, 6(5): 1063-1072.

[6] Villegas N M, Sánchez C, Díaz-Cely J, et al. Characterizing context-aware recommender systems: A systematic literature review. Knowledge-Based Systems, 2018, 140: 173-200..

[7] Yang X, Guo Y, Liu Y, et al. A survey of collaborative filtering based social recommender systems. Computer communications, 2014, 41: 1-10.

[8] Shih Y Y, Liu D R. Hybrid recommendation approaches: collaborative filtering via valuable content information. Proceedings of the 38th Annual Hawaii International Conference on System Sciences. IEEE, 2005, 217b-217b.

[9] Qi L, Xu X, Dou W, et al. Time-aware IoE service recommendation on sparse data. Mobile Information Systems, 2016.

[10] Zhang X, Qi L, Dou W, et al. MRMondrian: scalable multidimensional anonymisation for big data privacy preservation. IEEE Transactions on Big Data, 2017.

[11] Dou W, Qi L, Zhang X, et al. An evaluation method of outsourcing services for developing an elastic cloud platform. The Journal of Supercomputing, 2013, 63(1): 1-23.

[12] Xu X, Li Y, Huang T, et al. An energy-aware computation offloading method for smart edge computing in wireless metropolitan area networks. Journal of Network and Computer Applications, 2019, 133: 75-85.

[13] Zheng V W, Cao B, Zheng Y, et al. Collaborative filtering meets mobile recommendation: A user-centered approach. Twenty-Fourth AAAI Conference on Artificial Intelligence. 2010.

[14] Truong K Q, Ishikawa F, Honiden S. Improving accuracy of recommender system by item clustering. IEICE TRANSACTIONS on Information and Systems, 2007, 90(9): 1363-1373.

[15] Hong W, Zheng S, Wang H, et al. A job recommender system based on user clustering. JCP, 2013, 8(8): 1960-1967.

[16] Shu L. User Clustering Topic Recommendation Algorithm based on Two Phase in the Social Network. International Journal of Intelligent Information and Management Science, 2017 6(3):5-11.

[17] Ghazanfar M A, Prugel A. The advantage of careful imputation sources in sparse data-environment of recommender systems: Generating improved svd-based recommendations. Informatica, 2013, 37(1).

[18] Meng S, Qi L, Li Q, et al. Privacy-preserving and sparsity-aware location-based prediction method for collaborative recommender systems. Future Generation Computer Systems, 2019, 96: 324-335.

[19] Xu Z, Gu R, Huang T, et al. An IoT-oriented offloading method with privacy preservation for cloudlet-enabled wireless metropolitan area networks. Sensors, 2018, 18(9): 3030.

[20] Herlocker J L, konstan J A, Borchers a, et al. An algorithm framework for performing collaborative filtering. ACM SIGIR Conference on research and development in information retrieval, Berkeley, 1999:230-235.

[21] Xia Z, Zhu Y, Sun X, et al. Towards privacy-preserving content-based image retrieval in cloud computing. IEEE Transactions on Cloud Computing, 2015, 6(1): 276-286.

[22] Breese JS, Heckerman D, et al. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. International Conference on Uncertainty in Artificial Intelligence. 1998,43-52.

[23] Linden G, Smith B, York J. Amazon. com recommendations: Item-to-item collaborative filtering. IEEE Internet computing, 2003, 7(1): 76-80.

[24] Polatidis N, Georgiadis C K, Pimenidis E, et al. Privacy-preserving collaborative recommendations based on random perturbations. Expert Systems with Applications, 2017, 71: 18-25.

[25] Li Y, Liu S, Wang J, et al. A local-clustering-based personalized differential privacy framework for user-based collaborative filtering. International Conference on Database Systems for Advanced Applications. Springer, Cham, 2017: 543-558.

[26] Keshavarz S, Honarvar A R. A Parallel Paper recommender system in Big Data Scholarly. International Conference on Electrical Engineering and Computer. 2015.

[27] Zheng Y, Xiao L, Tang W, et al. A Music Recommendation Method for Large-Scale Music Library on a Heterogeneous Platform. International Conference on Algorithms and Architectures for Parallel Processing. Springer, Cham, 2014: 472-482.

[28] Liu D, Huo C, Yan H. Research of commodity recommendation workflow based on LSH algorithm. Multimedia Tools and Applications, 2019, 78(4): 4327-4345.

[29] Yan B, Yu J, Yang M, et al. A novel distributed Social Internet of Things service recommendation scheme based on LSH forest. Personal and Ubiquitous Computing, 2019: 1-14.

[30] Qi L, Zhang X, Dou W, et al. A two-stage locality-sensitive hashing based approach for privacy-preserving mobile service recommendation in cross-platform edge environment. Future Generation Computer Systems, 2018, 88: 636-643.