Knowledge-Based Systems 204 (2020) 106196

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Diversified service recommendation with high accuracy and efficiency

Lina Wang[a], Xuyun Zhang[b], Ruili Wang[c], Chao Yan[a], Huaizhen Kou[a], Lianyong Qi[a,d,*]

[a] School of Information Science and Engineering, Qufu Normal University, Rizhao, China
[b] Department of Computing, Macquarie University, Australia
[c] Institute of Natural and Mathematical Sciences, Massey University, New Zealand
[d] State Key Laboratory for Novel Software Technology, Nanjing University, China

ARTICLE INFO

Article history:
Received 6 November 2019
Received in revised form 1 June 2020
Accepted 26 June 2020
Available online 3 July 2020

Keywords:
Recommender system
Collaborative Filtering
Accuracy
Diversity
Efficiency

ABSTRACT

Collaborative filtering-based recommender systems are regarded as an important tool to predict the items that users will appreciate based on the historical usage of users. However, traditional recommendation solutions often pay more attentions to the accuracy of the recommended items while neglect the diversity of the final recommended list, which may produce partial redundant items in the recommended list and as a result, decrease the satisfaction degree of users. Moreover, historical usage data for recommendation decision-making often update frequently, which may lead to low recommendation efficiency as well as scalability especially in the big data environment. Considering these drawbacks, a novel method called *DivRec_LSH* is proposed in this paper to achieve diversified and efficient recommendations, which is based on the historical usage records and the Locality-Sensitive Hashing (LSH) technique. Finally, we compare our method with existing methods on the *MovieLens* dataset. Experiment results indicate that our proposal is feasible in addressing the triple dilemmas of recommender systems simultaneously, i.e., high efficiency, accuracy and diversity.

© 2020 Elsevier B.V. All rights reserved.

1. Introduction

With the advent of the big data age, people are confronted with a big volume of service information resources, which often makes it difficult to quickly find the valuable information that users are interested in [1–4]. In this situation, service recommendation (e.g., movie recommendation, music recommendation, book recommendation, etc.) has become an effective way to quickly extract insightful information from massive data, and has been widely adopted in many big data applications, such as Amazon.com, Facebook.com, and Netflix.com [5,6]. Traditional collaborative filtering (CF)-based recommender systems typically try to mine user preferences based on user–item usage records, and then automatically recommend a set of appropriate items that match a user's tastes most [7–10].

Traditional recommendation methods based on CF mainly lay emphasis on the improvement made to recommendation accuracy, which probably cause redundant items in the limited Top-N recommended list and as a result, the user satisfaction may be compromised. In this case, it is desirable for a recommender system to provide users with appropriate diversified items. Diversified recommendation can avoid redundancy and enhance the user's range of choices, which is conducive to resolving the uncertainty in the prediction of users' preference. However, for a recommender system, it is often facing the dilemma between accuracy and diversity. Despite the possibility of achieving high accuracy by recommending the most popular items to users, a loss of diversity often results. On the contrary, high diversity can be realized by offering personalized items for each user, which may result in a decline in accuracy. Therefore, it is vital for a recommender system to provide optimal recommended list with a balance between diversity and accuracy.

Moreover, traditional CF-based approaches, e.g., neighborhood-based CF, often investigate every user (or item) in the dataset to identify K-nearest neighbors, which has a potential to cause low recommendation efficiency, especially under the condition where the volumes of the dataset get increasingly high with updates over time.

In view of these challenges, we boost the existing CF-based recommendation method through combining the technique of Locality-Sensitive Hashing (LSH), a kind of efficient search technique. Moreover, a new recommendation method called *DivRec_LSH* (Diversified Recommendation based on LSH) is proposed, which can be divided into two parts, finding similar items and diversifying recommended list. Benefiting from LSH principles and its inherent advantages in efficiency, *DivRec_LSH*

* Corresponding author at: School of Information Science and Engineering, Qufu Normal University, Rizhao, China.
E-mail addresses: linawang425@163.com (L. Wang), xuyun.zhang@mq.edu.au (X. Zhang), ruili.wang@massey.ac.nz (R. Wang), yanchao@qfnu.edu.cn (C. Yan), kouhuaizhen@hotmail.com (H. Kou), lianyongqi@gmail.com (L. Qi).

https://doi.org/10.1016/j.knosys.2020.106196
0950-7051/© 2020 Elsevier B.V. All rights reserved.

can provide accurate, diversified, and efficient service recommendations (e.g., movie recommendation or book recommendation).

The main contributions in this paper are as follows:

(1) We employ LSH technique to handle the problem of excessive time cost in traditional CF-based recommendation approaches.

(2) A novel diversified method based on LSH is developed in this paper, which can find the optimal recommended list that not only meets users' interests but also maintain a sound diversity.

(3) Comprehensive experiments are executed using *MovieLens* dataset to prove the advantages of our proposal. Experiment results indicate that our proposal is feasible in addressing the triple dilemmas of recommender systems simultaneously, i.e., high efficiency, accuracy and diversity.

The remainder of the paper is organized as follows. Related work is summarized in Section 2. In Section 3, we demonstrate the motivation of this paper. In Section 4, we first describe the LSH technique briefly. Then, the proposed *DivRec_LSH* method is presented in detail. In order to prove the effectiveness of our method compared to other methods, comprehensive experiments are executed in Section 5. At last, in Section 6, we conclude the paper and point out our improvement directions in the future.

2. Related work

2.1. Accuracy-oriented recommendation

In recent years, many researchers have focused on exploring new methods that suitable for improving the accuracy of recommender systems [11,12]. In [13], Liu et al. conducted analysis of various methods for the calculation of similarity, and put forward a novel approach to enhance the traditional CF algorithm. In [14], Fan et al. combined user activity level with content-based recommendation algorithm to make prediction of the empty values in user–item rating matrix while calculating similarity between users. In [15], Jia et al. used a user-based CF approach with double neighbor selecting tactics. When choosing neighbors, similarity and trust value among users were taken into consideration. It can be seen from the experiment results that this approach can effectively improve recommendation accuracy. In [16], Xiao et al. considered that the user behaviors have the time sequence characteristic, and proposed an improved user-based CF recommendation method, which took time information into account at the time of computation of user similarity. In [17], Liu et al. made use of location information of users when searching for neighbors, and proposed an improved similarity measurement for enhancement to recommendation accuracy. In addition, many researchers start to take trust and weighted applications into consideration to enhance accuracy of the recommendation algorithm [18–20].

2.2. Diversity-oriented recommendation

To avoid redundancy of recommended results, diversity recommendation is raised, which represents that items in the recommended list not only satisfy interests of target users, but also cannot be overly similar to each other. Diversified recommendation approaches are capable to produce more personalized results. Therefore, in recent years, it has attracted extensive attention from industry and academia. In [21], Mouzhi Ge et. categorized diversity into two types, namely individual diversity and aggregate diversity. Individual diversity usually measures the average difference between all item pairs in each individual user's recommended list, whereas aggregate diversity means the total number of different items in all recommended lists of all users. In this paper, our focus is placed on the individual diversity of recommendation.

In recent years, some methods for diversifying recommendation results have been put forward. Maximal Marginal Relevance (MMR) is a common approach of diversified ranking. In [22], Vargas et al. combined diversity and accuracy as the main objective of MMR to achieve a fine balance between them. In [23], Gan et al. applied an approach called Power Law Adjustments of User Similarities (PLUS) to improve personalize recommendation. In the framework of user-based CF, PLUS can adjust user similarity value by exercising power function to decrease the impact of popular items. In [24], Li et al. for large graphs, applied a new method of diversified ranking. The diversified ranking matter was treated as a function maximization problem of sub-modular set. In [25], Kyriak et al. proposed a method to calculate diversity of the system based on similarity between items and users. In [26], Tevfik Aytekin suggested a new method for combination of Clustering and CF, which can adjust the degree of diversity in the recommended lists for users. The core of this method is the establishment of cluster weights. In [27], Jiang et al. put forward a new approach known as ROUND, which merges relevance between users, relevance between items, relevance between items and users. They applied a strategy of K-nearest neighbor when filtering out unreliable connections in a network, and adopted a random walk model to describe the degree of correlations between user nodes and items nodes.

However, those above methods often require lots of computation cost, which has a potential to cause low recommendation efficiency. Moreover, while these methods all consider accuracy and diversity of recommended lists, there is still a dilemma between diversity and accuracy.

2.3. Efficiency-oriented recommendation

Traditional CF-based methods (e.g., item-based CF) may result in high computational complexity when K-nearest neighbors are selected, which could lead to low efficiency and scalability. Thus, to enhance recommendation scalability and efficiency, many researchers have developed various model-based CF techniques. Some researchers began to use the methods of matrix factorization [28–31]. Clustering-based methods [32] are also capable to improve the overall recommendation efficiency. Because in these methods, the recommendation process consists of two parts: model training and recommendations, and model training can be performed offline. However, as users' historical data changes over time, the model needs to be updated frequently. Thus, these methods also show poor efficiency, which may make it impossible for them to meet the quick response requirements of users.

Much of the work tries to enhance diversity and accuracy of recommendations. But our proposal attempts to address the triple dilemmas of recommender systems simultaneously, i.e., high efficiency, accuracy and diversity.

3. Motivation

Here, we utilize the example in Fig. 1 to demonstrate the motivation of our proposal. Assumed that there is a planform, Netflix in Fig. 1, a large number of movies in Netflix, such as movie *'Green Book'*, *'The Shawshank Redemption'*, *'Forrest Gump'*, *'Léon'*, *'Joker'*, *'Ne Zha'*, *'Forever Young'*, among of which movies *'Green Book'*, *'The Shawshank Redemption'*, *'Forrest Gump'* and *'Léon'* have been overrated by Tom. Now, Netflix intends to recommend its own movies to Tom. In the light of the traditional CF method, i.e., item-based CF, the first step is to find movies that are similar to the historical records of Tom. The second step is a recommended list generation where the movies are selected from the candidate movies, i.e., movies *'Ne Zha'*, *'Before Midnight'*, ..., *'Joker'* form

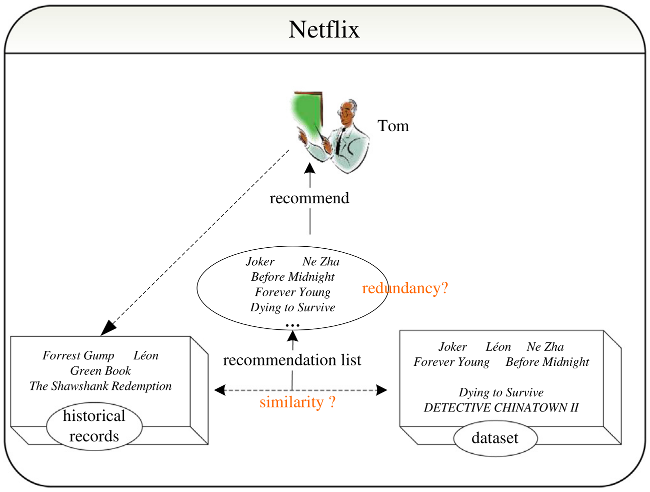


Fig. 1. CF-based recommendation: an example.

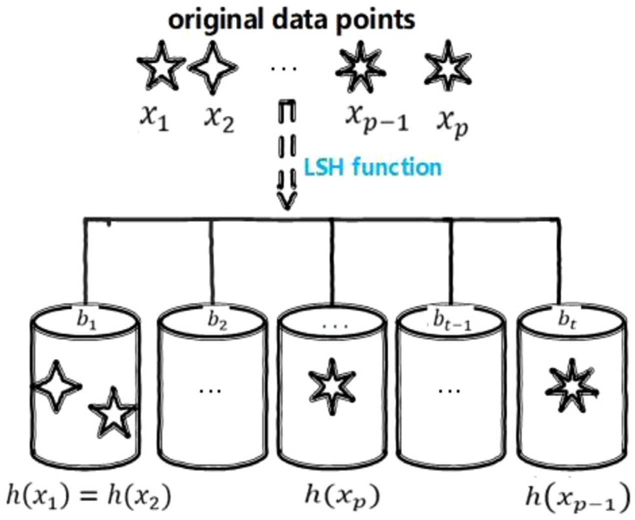


Fig. 2. The process of LSH-based ANN search.

a recommended list. However, there are often two challenges arising from this as follows:

(1) The volume of data in Netflix has a potential to update on a frequent basis over time, which requires frequent calculation of item similarity, as a result of which recommendation efficiency could be compromised to the extent that the quick response requirements of users cannot be met.

(2) As the diversity of recommended lists is neglected, the items in list may be similar to each other, which often lead to redundancy and failure to meet each user's wide interests and finally reduce the user satisfaction.

Considering the above issues, we put forward a novel recommendation method, named *DivRec_LSH*, to achieve accurate, diversified and efficient recommendation goals.

4. The proposed recommendation method

In Section 4.1, the LSH technique is introduced briefly. Then, we present the details of our proposed method *DivRec_LSH* in Section 4.2.

4.1. Locality-Sensitive Hashing

In neighbor-based CF, we need to find K-nearest neighbors of users when making prediction of rating. Nevertheless, traditional neighbor-based CF algorithms require a search to be conducted of all users (or items) in the datasets for identification of the nearest neighbors, which is usually not suitable for constant-changing datasets well. Locality-Sensitive Hashing, one of the approximate nearest neighbor (ANN) search methods, has been validated as capable to resolve this problem. The idea of LSH is to classify items using locality-sensitive functions, for which similar or close items have more chance of being hashed into the same bucket than different items do.

According to [33], LSH function should satisfy both conditions (1) and (2). Here, m and n represent the original data points; $h(m)$ and $h(n)$ are the hashing values of m and n respectively; the distance between m and n is $d(m,n)$; $P(X)$ is the probability of event X holds; p_1 and p_2 represent the probability threshold; d_1 and d_2 represent the distance threshold. LSH function $h(\cdot)$ satisfy the following conditions:

$$\text{If } d(m,n) \le d_1, \text{ then } P(h(m) = h(n)) \ge p_1 \tag{1}$$

$$\text{If } d(m,n) \ge d_2, \text{ then } P(h(m) = h(n)) \le p_2 \tag{2}$$

where $p_1 > p_2$ holds.

In the scheme of LSH, LSH functions can decide whether or not two items are a similar item. Suppose that data points a and b are hashed by LSH function to get the hash values $h(a)$ and $h(b)$ respectively. The notation "$h(a) = h(b)$" means that "a and b are a similar pair", and the notation "$h(a) \neq h(b)$" means that "a and b are not a similar pair".

We use Fig. 2 to illustrate the core of the LSH more intuitively. Suppose that $x_1, x_2, \ldots, x_p$ represent the original data points. Then through LSH functions, $x_1, x_2, \ldots, x_p$ are mapped to buckets $b_1, b_2, \ldots, b_t$. According to the idea of LSH, the points x_1 and x_2 that are close to each other, so they can be projected into same bucket b_1, while the points x_{p-1} and x_p that are not close to each other are mapped into different bucket. Then, based on the notation $h(x_1) = h(x_2)$, we can say that x_1 and x_2 are similar in high probability. On the contrary, the notation $h(x_{p-1}) \neq h(x_p)$ represents that x_{p-1} and x_p are not a similar pair. Namely, items in the same bucket can be considered similar, while items in different buckets can be considered different.

As indicated by the above-mentioned example, the original data points $x_1, x_2, \ldots, x_p$ can be placed in a single LSH table, and the LSH table can be built offline, which is beneficial to enhance the efficiency of approximate nearest neighbor search. In this paper, we apply LSH to neighbor-based CF to further make diversified recommendations.

4.2. DivRec_LSH: Diversified Recommendation based on LSH

In this part, we put forward a new recommendation method named *DivRec_LSH*. The basic idea of our proposal is: based on LSH technique, we first transform rating data into item indices; afterward, we utilize the item indices to find items similar to the target user history; then, we predict the missing rating; finally, according to item indices, we select items with different indices to form a list and recommend. Concretely, as is shown in Fig. 3, the method consists of four steps. The symbols mentioned in this paper and their meanings are shown in Table 1.

Step 1: Build item index offline

In this step, we build index of items to achieve efficient recommendation. As specified in Table 1, $x_1, x_2, \ldots, x_p$ denote all items in dataset. Here, we select an LSH function family $H(\cdot)$ to construct index for every item in the dataset offline, while the 'distance' type of $d(\cdot)$ (see Section 4.1) decides the LSH function

Step 1: Build item index offline

For each item i in dataset, according to $i's$ observed user rating data, build $i's$ index $H(i)$ based on an LSH function family $H(\cdot) = \{h_1(\cdot), \dots, h_r(\cdot)\}$.

Step 2: Establish a set of similar items

For a target user U^*, extract his/her historical records and build history set, according to the selected LSH function family $H(\cdot) = \{h_1(\cdot), \dots, h_r(\cdot)\}$, build item-index for each item j in the history set. Then, we look for the item j's qualified neighboring items and put them into set sim_set.

Step 3: Rating prediction

According to U^*'s Sim_Set obtained in Step 2, we select the items never rated by U^* in Sim_Set and predict their ratings by U^*.

Step 4: Diversified recommended list generation

Sort items in descending order according to their predicted ratings by U^* based on Step 3. Then we select k (the length of the recommended list) items to form an accurate and diversified list to recommend to U^*.

Fig. 3. Four steps of the *DivRec_LSH* method.

family $H(\cdot)$. In CF-based recommendation, PCC is often used for distance measurement. Therefore, we adopt LSH function $h(\cdot)$ corresponding to PCC to establish item index.

First, for an item $x_i (1 \le i \le p)$, we can model its' rating data over n users, i.e., $x_{i,u1}$, $x_{i,u2}$, ..., $x_{i,un}$, with an n-dimensional vector $\vec{x_i} = (x_{i,u1}, x_{i,u2}, \dots, x_{i,un})$, where $x_{i,uk} = 0$ if the user u_k has never rated the item x_i. Then, for vector $\vec{x_i}$, the hash value $h(\vec{x_i})$ can be calculated by the LSH function in (3) [34]. Here, $\vec{v}$ is a vector $(v_1, v_2, \dots, v_n)$ with n-dimensional, where v_i $(1 \le i \le n)$ can be chosen from $[-1,1]$ randomly. In order to make it easy for readers to understand, the physical meaning of (3) is explained as below: vector $\vec{v}$ can be regard as hyperplane of space division: suppose that vector $\vec{i_s}$ and vector $\vec{i_q}$ $(1 \le s, q \le p, s \ne q)$ are on the same side of $\vec{v}$ (i.e., $\vec{i_s} \circ \vec{v} > 0$, $\vec{i_q} \circ \vec{v} > 0$ hold; or $\vec{i_s} \circ \vec{v} \le 0$, $\vec{i_q} \circ \vec{v} \le 0$ hold; the symbol "$\circ$" means the inner product between two vectors according to the equation in (4), then $\vec{i_s}$ and $\vec{i_q}$ can be considered as similar neighbor (with a high possibility).

$$h(i) = \begin{cases} 1 & \vec{x_i} \circ \vec{v} > 0 \\ 0 & \vec{x_i} \circ \vec{v} \le 0 \end{cases} \tag{3}$$

$$\vec{x_i} \circ \vec{v} = \sum_{k=1}^{n} x_{i,uk} \cdot v_k \tag{4}$$

Therefore, after the hash map in (3), n-dimensional vector $\vec{x_i}$ is transformed into one Boolean value. However, LSH is a search method based on probability, the "similarity preserving" property of LSH cannot be reflected by only one LSH function. Based on this, more LSH functions are adopted here. We randomly choose r LSH functions $h_1(\cdot), h_2(\cdot), \dots, h_r(\cdot)$ to generate an r-dimensional 0–1 vector $H(\vec{x_i}) = (h_1(\vec{x_i}), \dots, h_r(\vec{x_i}))$, where $h_m(\vec{x_i})$ $(1 \le m \le r)$ can be calculated by (3). Then $H(\vec{x_i})$ can be regarded as the index of item x_i. According to the indices $H(\vec{x_1}), H(\vec{x_2}), \dots, H(\vec{x_p})$, we can generate an LSH table *h_Table*, the LSH table *h_Table* records the mapping relationships of "$x_i \to H(\vec{x_i})$".

Based on the above discussion, the items indices can be built offline. We can represent the pseudo-code of this step with Algorithm 1.

Algorithm 1: Build item index offline

Inputs: $I = \{x_1, \dots, x_p\}$: item set
$U = \{u_1, \dots, u_n\}$: user set
User-item rating matrix
Outputs: an LSH table h_Table with buckets

```
1.  for $j$ = 1 to $r$ do // $r$ LSH functions in an LSH table
2.      for $k$ = 1 to $n$ do // generate an $n$-dimension vector
3.          $v_{jk} = random[-1,1]$
4.      end for
5.      $\vec{v_j} = (v_{j1}, \dots, v_{jn})$
6.      for $a$ = 1 to $p$ do // $p$ items
7.          $\vec{x_a} = (x_{a,u1}, \dots, x_{a,un})$
8.          if $\vec{x_a} \circ \vec{v_j} > 0$
9.              then $h_j(\vec{x_a}) = 1$
10.             else $h_j(\vec{x_a}) = 0$
11.         end if
12.     end for
13. end for
14. for $a$ = 1 to $p$ do
15.     $H(\vec{x_a}) = (h_1(\vec{x_a}), \dots, h_r(\vec{x_a}))$
16. end for
17. build an LSH table $h\_Table$ according to $H(\vec{x_1}), H(\vec{x_2}), \dots, H(\vec{x_p})$
18. return LSH table $h\_Table$
```

Step 2: Establish a set of similar items

In Step 1, we have obtained the LSH table *h_Table*. Therefore, according to the scheme of LSH, if two items x_1 and x_2 have the same index value in the LSH table *h_Table*, i.e., $H(\vec{x_1}) = H(\vec{x_2})$ holds in *h_Table*, then x_1 and x_2 are similar items with a high probability. In this way, the items that are similar to the user U^* historical records can be identified.

First, we extract historical records of U^* and put them into history set *h_set*. According to the LSH function family $H(\cdot)$ chosen in Step 1, each item j in the *h_set* index $H(j)$ is calculated online and an LSH table *his_H_Table* is built based on $H(j)$. As specified in Table 1, $b_1, b_2, \dots, b_t$ denote buckets in an LSH table, and the index number of the item in the bucket indicates the bucket number. Then, the bucket b_s $(1 \le s \le t)$ that appears in *h_Table* and *his_H_Table* is selected, and items in b_s can be put into the similar set *sim_set*. There may be two cases. One is that there may be multiple buckets that satisfy the condition, for which there

Table 1
Meanings of the symbols in this paper.

Symbol	Meaning
U^*	A target user
n	Number of users
$H(\cdot)$	An LSH function family
$P(X)$	The probability of event X holds
$d(\cdot)$	Distance between two original data points (or items)
β	The ratio of removing partial entries from the user–item rating matrix
d_1, d_2	Distance threshold
p_1, p_2	Probability threshold
$x_1, x_2, \ldots, x_p$	Original data points (items in dataset)
$H\_Table_1, \ldots, H\_Table_T$	T LSH tables
$h_1(\cdot), \ldots, h_r(\cdot)$	r LSH functions
$b_1, b_2, \ldots, b_t$	t buckets in an LSH table

are multiple *sim_sets* in this case. The other one is that there is no bucket that fits the condition. Despite this, it cannot be assumed that there is no item that is similar to the U^*'s historical records because LSH is a probability-based approximate method. In this case, rather than creating only one LSH table, repeat Step 1 and Step 2 to build T *h_Tables*: $h\_Table_1, \ldots, h\_Table_T$ and T *his_H_Tables*: $his\_H\_Table_1, \ldots, his\_H\_Table_T$. If the condition in (5) holds, we can put items in $bucket_j$ into the similar set *sim_set*. At last, we put all sets *sim_set* into set *Sim_Set*.

$$\exists i \in \{1, \ldots, T\} \text{ and } j \in \{1, \ldots, t\} \tag{5}$$

satisfy $bucket_j$ appears in h_Table_i and $his\_H\_Table_i$

In this paper, we do not consider the appearance times of similar items. The pseudo-code of this step can be described by Algorithm 2.

Algorithm 2: Establish a set of similar items

Inputs: U^*: a target user
h_set: the history records of U^*
Outputs: Sim_Set

```
1.  for x = 1 to T do // T LSH tables
2.      repeat Algorithm 1 create $h\_Table_x$
3.      for s = 1 to b do // b items in $h\_set$
4.          according to Algorithm 1, calculate $H(\vec{\iota_s})$
5.      end for
6.      build LSH table $his\_H\_Table_x$ according to $H(\vec{\iota_1}), \ldots, H(\vec{\iota_b})$
7.      find bucket b that appears in $h\_Table_x$ and $his\_H\_Table_x$
8.      If bucket $b \neq$ Null
9.          $sim\_set = \emptyset$
10.         then place items in b in LSH table $h\_Table_x$ into $sim\_set$, and place
                items in b in hash table $his\_H\_Table_x$ into $sim\_set$
11.     end if
12.     put $sim\_set$ into set $Sim\_Set$
13. end for
14. return $Sim\_Set$
```

Step 3: Rating prediction

For each item i never rated by U^* before, its future rating by U^* can be predicted based on U^* historical records. According to Step 2, we have gained the set *Sim_Set*. For each set *sim_set* in *Sim_Set*, we search for item i that U^* has never rated, and find the item m that appears in the same set *sim_set* as item i but has been overrated by U^*. Then we can obtain predictive raring for item i in (6). The pseudo-code of this step is specified more formally by Algorithm 3.

$$Score_i = \frac{\sum_{m \in sim\_set} rating_m}{total(m)} \tag{6}$$

where $rating_m$ represents the rating of item m that appears in the same set *sim_set* as item i but has been overrated by U^*. Here, m may come from multiple *sim_set*s, because i may appear in multiple *sim_sets*, $total(m)$ represents the number of item m.

Algorithm 3: Rating prediction

Inputs: Sim_Set
Outputs: the scores of items in Sim_Set that U^* has never rated

```
1.  for each $sim\_set$ in $Sim\_Set$ do
2.      find the item i that $U^*$ has never rated
3.      then put item i into set $predicted\_Set$
4.  end for
5.  for each item $i_f$ in $predicted\_Set$ do
6.      count = 0, sum = 0
7.      for each $sim\_set$ in $Sim\_Set$ do
8.          search the item j that $U^*$ has rated
9.          if item j and item $i_f$ appear in same $sim\_set$
10.             then count +=1, sum += $rating_j$
11.         end if
12.     end for
13.     if count > 0
14.         then $Score_{i_f}$ = sum / count
15.     end if
16.     return $Score_{i_f}$
17. end for
18. return $Score_{i_1}, Score_{i_2}, \ldots, Score_{i_f}$
```

Step 4: Diversified recommended list generation

In this step, we present a diversified recommended list. In order to offer users recommendation accuracy, a "ranking threshold w" is set in advance. First, according to the items' prediction rating obtained by Step 3, the items are sorted in descending order and the items with a scores below the threshold w are discounted. Then, according to the LSH rationale, if two items y_1 and y_2 have the different index value, namely $H(\vec{y_1}) \neq H(\vec{y_2})$, then y_1 and y_2 can be regarded as different items. Therefore, we select k (the size of recommended list) items with different index number (obtained in Step 1) to generate a new list, it may form multiple lists. Individual diversity means the average dissimilarity between each pair of items in a recommended list. We apply individual diversity to measure the diversity value of every list in (7). Last, we chose the list with highest individual diversity value to recommend to the target user U^*.

$$individual\ diversity = 1 - \frac{2}{N(N-1)} \sum_{i,j \in R(u), i \neq j} s(i,j) \tag{7}$$

where $R(u)$ represents the recommended list for U^* and $N = |R(u)|$. $s(i,j)$ denotes the similarity of items $i, j \in R(u)$. We evaluate the similarity between items using cosine similarity between the rating vectors of i and j.

The pseudo code of this step can be specified more formally by Algorithm 4.

Algorithm 4: Recommend a diversified list to U^*

Inputs: $Score_{i_1}, \ldots, Score_{i_f}$
k: the length of the list
w: ranking threshold
Outputs: a diversified list for U^*

```
1.  for $i_1, \ldots i_f$ do
2.      if $Score_{i_f} > w$
3.          then place $i_f$ into $i\_set$
4.      end if
5.  end for
6.  sort items in $i\_set$ in descending order according to their score
7.  for $t = 1$ to $h$ do // $h$ items in $i\_set$, where $h \leq f$
8.      select $k$ items with different index number to form a list
9.  end for
10. for each list do
11.     calculate diversity value by (6)
12. end for
13. return the list with highest diversity value to recommend to $U^*$
```

5. Experiments

5.1. Experimental settings

In this section, a set of experiments are tested using the *MovieLens* dataset (https://movielens.org/). This dataset contains ratings given by 6040 users for 3952 movies. The ratings range from 1 to 5 and unrated values are taken as zero. We randomly remove some items from the user–item rating matrix, so that we can predict the empty values and make further recommendation, the removal ratio is denoted by β. The experiments are run on a Dell PC with 2.2 GHz processors and 8.0 GB RAM. The machine runs under Windows 10 (64 bits) as well as Python 3.6. Each test is run 50 times, meanwhile their average experimental outcomes are used.

This paper uses four aspects to evaluate the recommendation performances:

(1) Time cost: time consumed can be used to measure recommendation efficiency.

(2) MAE (Mean Absolute Error): we use MAE to test predicted ratings' accuracy, as indicated in Eq. (8). The higher the MAE is, the smaller the recommendation accuracy would be.

$$MAE = \frac{\sum_{i \in R} |r'_{u,i} - r_{u,i}|}{|R|} \tag{8}$$

where $r'_{u,i}$ represents the predicted rating, $r_{u,i}$ means the actual rating, and $|R|$ is the number of items in the list.

(3) RMSE (Root Mean Squared Error): we use RMSE to measure the degree of difference between the real rating and the predicted rating, as indicated in Eq. (9). The higher the RMSE is, the smaller the recommendation accuracy would be.

$$RMSE = \sqrt{\frac{1}{|R|} \sum_{i=1}^{|R|} \left(r'_{u,i} - r_{u,i}\right)^2} \tag{9}$$

(4) *average_Div* (average of all *individual diversity*): we adopt the mean of individual diversity for all test users' recommended lists to evaluate the diversity of recommendation, as indicated in Eq. (10):

$$average\_Div = \frac{1}{U} \sum_{u \in U} (individual\ diversity) \tag{10}$$

where U represents the total number of test users.

5.2. Comparison methods

In order to measure the recommendation performances of our proposal, we compare it with the following four methods:

(1) *ClusDiv* [26]: it uses rating information of items to diversify the recommended list, and can be used after unknown ratings are predicted by the prediction method. Thus, we choose the variants of *ClusDiv*, i.e., *U_CF+ClusDiv*, *I_CF+ClusDiv* as the compared methods.
(2) *Entropy Fusion Approach (EF)* [35]: it first computes user exposure diversity and item concordance scores based on items' rating information, and then modifies the item similarity measures.
(3) *TDE_div* [36]: it uses two new hybrid ranking models to develop the final recommended list, one of which is a combination of the total diversity effect ranking and the standard ranking models. We choose this model as a compared method.
(4) *MF* [37]: it is a classic recommendation method. We choose this method as a baseline.

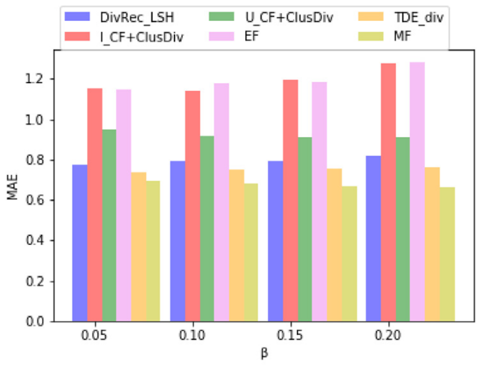
5.3. Experimental results and analyses

In this section, we test seven profiles to measure the performances of our method. Here, as described in Table 1, r denotes the number of LSH functions in an LSH table, T denotes the number of LSH tables.

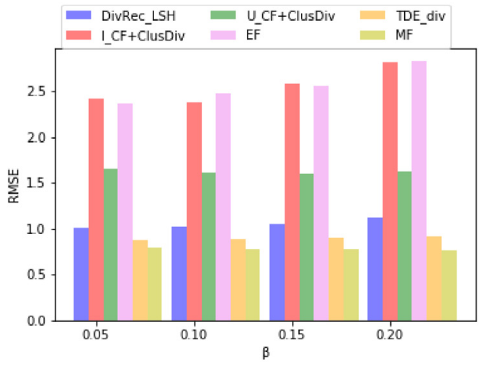
Profile 1: accuracy comparison of the six methods

In this part, we test and compare the accuracy of six methods by measuring the MAE and RMSE metrics. Here, $r = 4$, $T = 10$, β= {0.05, 0.10, 0.15, 0.2}, and ranking threshold $w = 3.5$, which means that the items whose predicted values are larger than w are taken as the candidate items. The experimental outcomes are indicated in Fig. 4.

As indicated in Fig. 4, *MF* method performs better than the other ones significantly in terms of both MAE and RMSE. This is due to the fact that matrix factorization technique can alleviate the data sparsity issue to some extent, which consequently improve the recommendation accuracy. Besides, although methods *DivRec_LSH*, *U_CF+ClusDiv*, *I_CF+ClusDiv*, *EF* and *TDE_div* all take account of both accuracy and diversity, *TDE_div* employs Tanimoto similarity to identify similar neighbors, which mitigates data sparsity to some degree. Therefore, *TDE_div* performs better than other methods. Compared to *MF* method, *TDE_div* takes diversity into consideration. As a result, *TDE_div* is less accurate than *MF*. Among the remaining four methods, *DivRec_LSH* performs the best in terms of accuracy. LSH is an approximate neighbor search method; therefore, most similar items of the item preferred by a user could be found according to LSH, which are more beneficial to item quality prediction. The performances



(a) MAE



(b) RMSE

Fig. 4. Recommendation accuracy comparison.

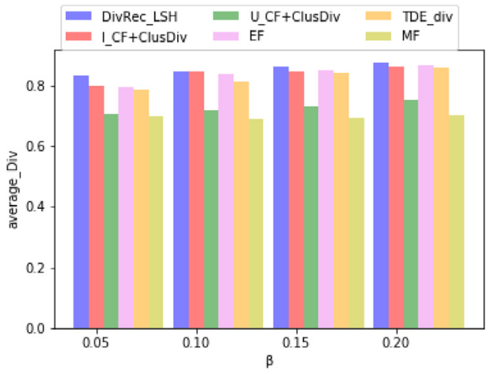


Fig. 5. Recommendation diversity comparison.

of *I_CF+ClusDiv* and *EF* are slightly inferior as they rely on item-based CF for item similarity calculation. While item-based CF is more suitable for finding long-tail items, which could compromise the accuracy of the entire recommender systems.

The values obtained from recommendation accuracy of the six methods all show a decreasing trend, while the value of β increases. This is for the reason that less recommendation information is available when the rating matrix becomes sparser.

Profile 2: diversity comparison of the six methods

In this part, we apply the *average_Div* metric to evaluate the diversity of six methods and compare results. Here, $r = 4$, $T = 10$, $\beta = \{0.05, 0.10, 0.15, 0.2\}$. The experimental outcomes are indicated in Fig. 5.

As indicated in Fig. 5, *MF* displays a worse performance in comparison with the other ones to a significant extent in terms

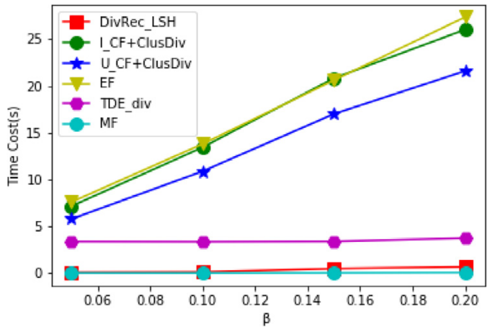


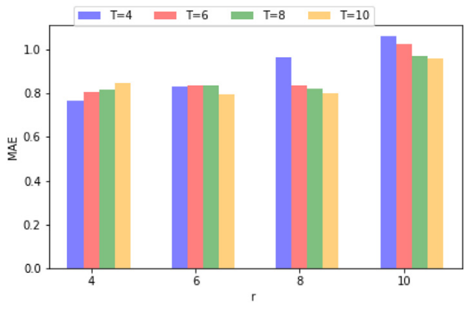
Fig. 6. Recommendation efficiency comparison.

of *average_Div*, which is because *MF* does not take into consideration the recommendation diversity. Among the remaining five methods, *DivRec_LSH* shows a better performance than the other four, including *U_CF+ClusDiv*, *I_CF+ClusDiv*, *EF*, *TDE_div*. This is caused by the reason that the items with different hash values are considered as dissimilar items based on LSH principle. Therefore, it is easy for us to filter out similar items by using the hash value of the item, so that it is probable that each pair of items in the list can ensure their dissimilarity to a certain degree. Under the condition that $\beta = 0.10$, the diversity values obtained from both *DivRec_LSH* and *I_CF+ClusDiv* are identical, but the accuracy value of *I_CF+ClusDiv* is lower in comparison with that of *DivRec_LSH* (see Fig. 4). In other words, *I_CF+ClusDiv* can improve the diversity of recommender results, but it is more likely that the results are unable to satisfy the users' interest. With regard to the *TDE_div* method, it holds the view that each item in a recommended list can impose its impact on the total diversity value of recommendations differently. Therefore, this method gives top priority to the items with high total diversity effect during the process of selection. Nevertheless, this is insufficient for the purpose of ensuring a high dissimilarity value between all item pairs in a recommended list. In addition, the performance shown by *U_CF+ClusDiv* is slightly inferior for the reason that it relies on user-based CF for the top-*N* list generation during the process of the first step. However, diversities have been available in user-based CF already, and thus updating the list in the second step does not help too much.

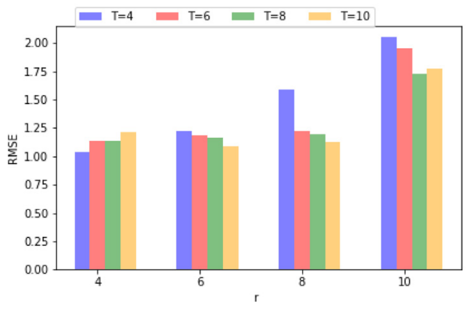
Profile 3: efficiency comparison of the six methods

In this section, we test and compare the efficiency of six methods by measuring the Time cost metric. Here, $r = 4$, $T = 10$, $\beta = \{0.05, 0.10, 0.15, 0.2\}$. The experimental outcomes are indicated in Fig. 6.

As displayed in Fig. 6, as β is on the rise, the time cost spent on all these six methods shows an increasing trend invariably. Behind which the rationale is that, the bigger β is, the more items are required in predicting rating. Consequently, the time required for similarity calculation increases as well. However, *EF* performs worse than other methods. Because *EF* involves the exposure diversity-related calculation of a target user. In comparison with the method *I_CF+ClusDiv*, *U_CF+ClusDiv* incurs less time cost, which is attributed to the reason that the dimension of the item is higher than the user dimension. As a result, less time cost is required in the operations related to similarity calculation in *U_CF+ClusDiv*. In *DivRec_LSH* and *MF*, most of the jobs could be finished offline, while the time cost spent on the remaining job (i.e., online similar neighbor search) in *MF* is almost $O(1)$. However, when the data related to user–item rating is updated along with the passage of time, the model of *MF* needs to be re-trained and updated on a frequent basis. Not similar to the method *MF*, *DivRec_LSH* only need to re-train the items that have



(a) MAE



(b) RMSE

Fig. 7. Recommendation accuracy comparison of *DivRec_LSH* w.r.t. (r, T).

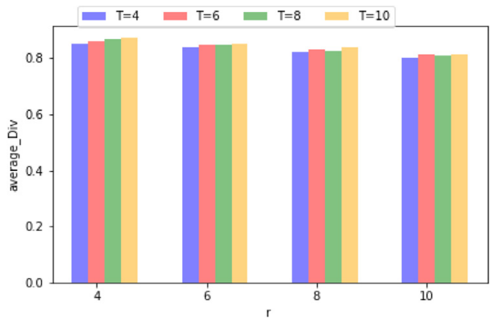


Fig. 8. Recommendation diversity comparison of *DivRec_LSH* w.r.t. (r, T).

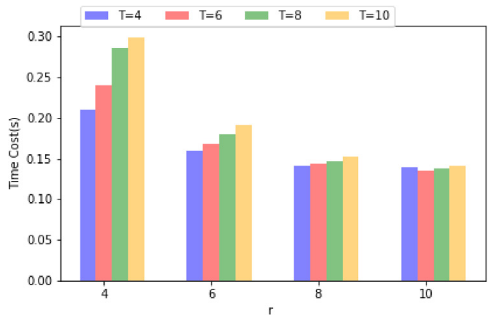


Fig. 9. Recommendation efficiency comparison of *DivRec_LSH* w.r.t. (r, T).

changed. Consequently, our proposed *DivRec_LSH* method can fit for the recommendation cases when a fast response from the user is in high demand.

Profile 4: Recommendation accuracy of *DivRec_LSH* w.r.t. (r, T)

In *DivRec_LSH*, the number of hash functions, i.e., r in an LSH table and the number of LSH tables, i.e., T play an essential role in making diversified, efficient and accurate recommendations. The more hash functions, the stricter the conditions to find the semblable neighbors for a target item. Therefore, the accuracy values of the proposed method with respect to r and T are tested through the MAE and RMSE metrics. Here, r and T are taken 4, 6, 8, 10 respectively, and $\beta = 0.2$. The experimental outcomes are indicated in Fig. 7.

As shown in Fig. 7, when T is 4 and r is on the rise, both MAE value and RMSE value show an increasing trend. Behind which the rationale is that, a larger r value implies more stringent for neighbor search, fewer qualified similar items of the target item can be searched and utilized for later item recommendations. In the case of the hash tables, i.e., T is 4 and the number of LSH functions, i.e., r is 10, both MAE value and RMSE value reach the maximum, which is because the condition of searching similar items is toughest so that target item has few similar items can be used recommendation decision-makings. When r is greater than 4 and T is rises, both MAE value and RMSE value show a decreasing trend. Because a larger T value indicates the condition of searching neighbors is looser, a few similar items of the target item be returned and made user of further item recommendation. Thus, recommendation accuracy will be improved.

Profile 5: Recommendation diversity of *DivRec_LSH* w.r.t. (r, T)

In this part, we measure the relationship between the recommended diversity of the proposed method and the parameters r and T. Here, $r = 4, 6, 8, 10$; $T = 4, 6, 8, 10$; $\beta = 0.2$. The experimental outcomes are indicated in Fig. 8.

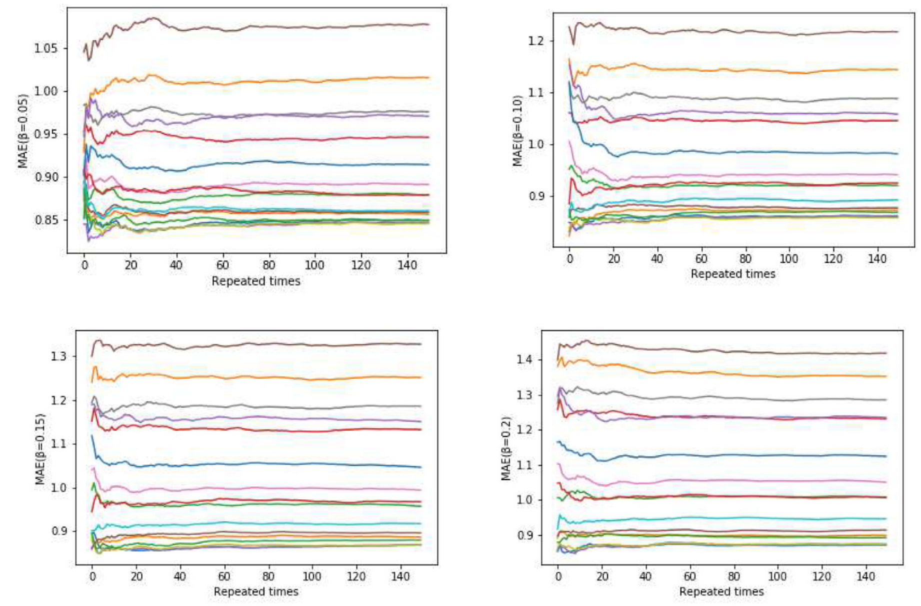
As Fig. 8 indicates, with r grows, the diversity values of *DivRec_LSH* decline, because the condition of searching similar items is stricter so that the partial similar items of target item may be overlooked and the results in similar items may presented in the recommended list. Another result that Fig. 8 indicates is that the diversity shows an upward trend when r is constant and T is on the increase, which is attributable to that, a larger T value indicates that the requirements of searching similar neighbors are relaxed, so that more similar items can be removed in Step 4 (see Section 4.2). In this case, the items in the recommended list show more diversity. In the case of $r = 4$ and $T = 10$, the value of diversity is the highest, because the condition of neighbor search is the least stringent, a few similar items in the list can be eliminated in Step 4 and correspondingly, items in the recommended list are not similar to each other with highly probability.

Profile 6: Recommendation efficiency of *DivRec_LSH* w.r.t. (r, T)

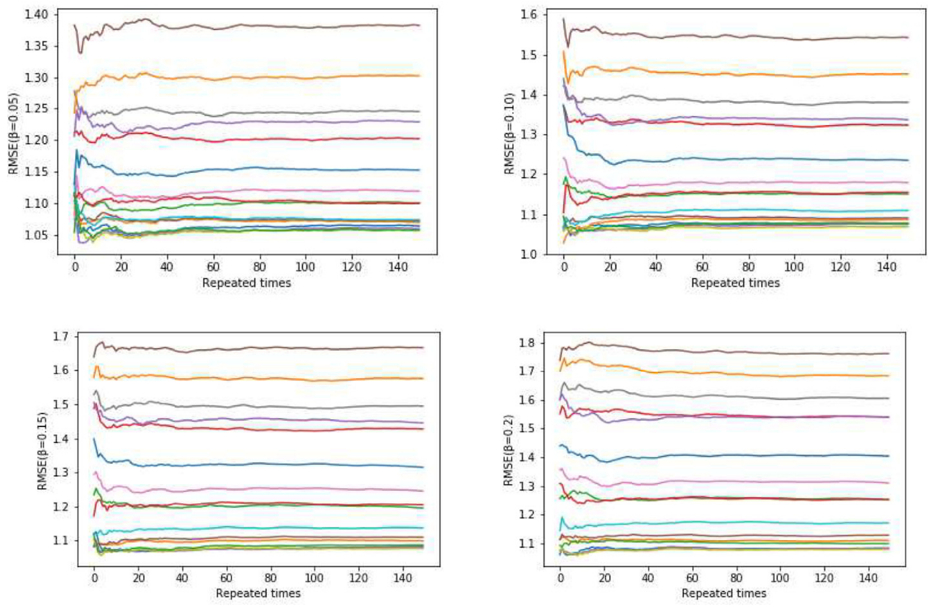
We evaluate the recommendation efficiency of our method *DivRec_LSH* with regard to r and T. Here, $r = 4, 6, 8, 10$; $T = 4, 6, 8, 10$; $\beta = 0.2$. The experiment outcomes are indicated in Fig. 9.

As shown in Fig. 9, we can see that the efficiency of our proposal shows an increasing trend when the number of hash functions, i.e., r rises. Because while r rises, the conditions of searching for similar neighbors gets more stringent. Accordingly, less similar neighbors of the target item are used to participate in further item recommendation; therefore, less computational time is required. Another result that Fig. 9 indicates is that the efficiency decreases when the number of LSH tables, i.e., T rises, which is because when T grows, the conditions of searching neighbors get looser, therefore, more similar neighbors of the target item can be found to utilize for further item recommendations; thus, more time cost is needed. When $r = 4$, $T = 10$, the time cost is highest, this result occurs because the conditions of neighbor search are the loosest, and more similar items of target item are returned and used to participate in further item recommendation.

Profile 7: The convergence of MAE and RMSE of *DivRec_LSH* w.r.t. experiment times



(a) MAE



(b) RMSE

Fig. 10. MAE/RMSE convergence of *DivRec_LSH* w.r.t. repeated times.

We test the convergence of MAE/RMSE of proposed approach *DivRec_LSH* with the experiment times. The number of experiments ranged from 1 to 150. The results are presented in Fig. 10. Here, each line in Fig. 10 represents an (r, T) parameter pair, which contains 16 parameter pairs, and r falls into {4, 6, 8, 10}; T belongs to {4, 6, 8, 10}; $\beta = 0.05, 0.10, 0.15, 0.2$.

As Fig. 10 shows, when the number of experiments grows, *DivRec_LSH* performs well in MAE and RMSE convergences. With each experiment repeated for more than 40 times, the MAE and RMSE values become relatively consistent. Hence, in our tests, each experiment was repeated 50 times, and their average outcomes were used finally.

5.4. Time complexity analyses

In this part, we analyze the time complexity of the proposed method. It is assumed that there are m items, n users, T LSH tables, g functions for each LSH table, and k items rated by a target user. Then the time costs incurred by the four steps in our proposal is detailed as below.

(1) In Step 1, item indices generation (including creating T LSH tables in Step 2) can be finished offline; therefore, the time complexity of this step is $O(1)$.

(2) In Step 2, firstly, the k item indices can be constructed, whose time complexity is $O(n*g*k)$. Afterward, we need to create T LSH tables, whose time cost is $O(n*g*k*T)$.

(3) In Step 3, the ratings of candidate items (at most m-k candidate items) are predicted on the basis of similar items (at most k items), whose time complexity is $O(k*(m\text{-}k))$.

(4) In Step 4, firstly, the items whose predicted ratings are larger than the given threshold are retained (at most m-k items). Then, the items with higher ratings and different indices are selected (at most m-k item indices are different) before a diversified recommended list is produced. Therefore, the time complexity in this Step is $O\,(m\text{-}k)$.

Through the above analyses, a conclusion could be drawn that our proposal's time complexity is $O\,(n*g*k*T+(m\text{-}k)*(k+1))$. The polynomial time complexity means that our method is usually time-efficient and scalable, as evidenced by the experimental results presented in Section 5.3.

6. Conclusion and future work

Diversity in recommendations is considered as an important dimension to interested users. Moreover, historical usage data for recommendation decision-makings may update frequently, which may lead to low recommendation efficiency. To deal with these problems, a novel approach *DivRec_LSH* is proposed to increase the diversity of recommendation lists, while achieving a high efficiency. Finally, a series of experiments are tested on *Movielens* dataset. As revealed by the experiments results, our proposal shows better with regard to diversity and efficiency while ensuring the accuracy of recommendation.

In this paper, we mainly exploit rating information about items to diversify the recommended lists, while neglect text information for items. Therefore, it may be insufficient to improve recommendation performances. In the future, we will continue improving the diversity-aware recommendation model by further considering users' context information (e.g., time and location) and items' context information (e.g., genre), so as to make more personalized recommendations.

CRediT authorship contribution statement

Lina Wang: Conceptualization, Methodology, Software. **Xuyun Zhang:** Methodology. **Ruili Wang:** Writing - review & editing. **Chao Yan:** Validation, Visualization. **Huaizhen Kou:** Software, Validation. **Lianyong Qi:** Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by the National Key Research and Development Program of China (No. 2017YFB1001800), the National Natural Science Foundation of China under Grant No. 61872219, the Natural Science Foundation of Shandong Province (ZR2019MF001), Open Project of State Key Laboratory for Novel Software Technology (No. KFKT2020B08).

References

[1] Y. Zhang, Y. Zhou, F. Wang, Service recommendation based on quotient space granularity analysis and covering algorithm on spark, Knowl.-Based Syst. 147 (2018) 25–35.
[2] J. Li, T. Cai, K. Deng, X. Wang, T. Sellis, F. Xia, Community-diversified influence maximization in social networks, Inf. Syst. 92 (2020) 1–12.
[3] H. Liu, H. Kou, C. Yan, L. Qi, Keywords-driven and popularity-aware paper recommendation based on undirected paper citation graph, Complexity 2020 (2020) 2085638, 15 p.
[4] L. Qi, Q. He, F. Chen, X. Zhang, W. Dou, Q. Ni, Data-driven web APIs recommendation for building web applications, IEEE Trans. Big Data (2020) http://dx.doi.org/10.1109/TBDATA.2020.2975587.
[5] S. Forouzandeh, A.R. Aghdan, Health recommender system in social networks: a case of facebook, Webology 16 (1) (2019).
[6] T. Cai, J. Li, A.S. Mian, R. Li, T. Sellis, J.X. Yu, Target-aware holistic influence maximization in spatial social networks, IEEE Trans. Knowl. Data Eng. (2020) http://dx.doi.org/10.1109/TKDE.2020.3003047.
[7] W. Zhong, X. Yin, X. Zhang, S. Li, W. Dou, R. Wang, L. Qi, Multi-dimensional quality-driven service recommendation with privacy-preservation in mobile edge environment, Comput. Commun. (2020) http://dx.doi.org/10.1016/j.comcom.2020.04.018.
[8] X. Chi, C. Yan, H. Wang, W. Rafique, L. Qi, Amplified LSH-based recommender systems with privacy protection, Concurr. Comput.: Pract. Exper. (2020) http://dx.doi.org/10.1002/CPE.5681.
[9] X. Wang, L.T. Yang, L. Kuang, X. Liu, Q. Zhang, M. Jamal Deen, A tensor-based big data-driven routing recommendation approach for heterogeneous networks, IEEE Netw. Mag. 33 (1) (2019) 64–69.
[10] L. Qi, X. Wang, X. Xu, W. Dou, S. Li, Privacy-aware cross-platform service recommendation based on enhanced locality-sensitive hashing, IEEE Trans. Netw. Sci. Eng. (2020) http://dx.doi.org/10.1109/TNSE.2020.2969489.
[11] Q. Zhang, D. Wu, J. Lu, F. Liu, G. Zhang, A cross-domain recommender system with consistent information transfer, Decis. Support Syst. 104 (2017) 49–63.
[12] S. Forouzandeh, A. Aghdam, S. Xu, S. Forouzandeh, Addressing the cold-start problem using data mining techniques and improving recommender systems by cuckoo algorithm: a case study of facebook, IEEE Comput. Sci. Eng. (2018).
[13] H. Liu, Z. Hu, A. Mian, H. Tian, X. Zhu, A new user similarity model to improve the accuracy of collaborative filtering, Knowl.-Based Syst. 56 (2014) 156–166.
[14] J. Fan, W. Pan, L. Jiang, A improved collaborative filtering algorithm combining content-based algorithm and user activity, in: Proc. of IEEE International Conference on Big Data and Smart Computing, 2014, pp. 88–91.
[15] D. Jia, F. Zhang, A collaborative filtering recommendation algorithm based on double neighbor choosing strategy, Comput. Res. Dev. 50 (5) (2013) 1076–1084.
[16] Y. Xiao, Pengqiang. A.I., Hsu. C.H., J. Xu, Time-ordered collaborative filtering for news recommendation, China Commun. 12 (12) (2015) 53–62.
[17] J. Liu, M. Tang, Z. Zheng, X. Liu, Location-aware and personalized collaborative filtering for web service recommendation, IEEE Trans. Serv. Comput. 9 (5) (2015) 686–699.
[18] C. Zhou, A. Li, A. Hou, Z. Zhang, Z. Zhang, F. Wang, Modeling methodology for early warning of chronic heart failure based on real medical big data, Expert Syst. Appl. (2020) http://dx.doi.org/10.1016/j.eswa.2020.113361.
[19] Y. Wang, Z. Cai, Z. Zhan, Y. Gong, X. Tong, An optimization and auction-based incentive mechanism to maximize social welfare for mobile crowdsourcing, IEEE Trans. Comput. Soc. Syst. 6 (3) (2019) 414–429.
[20] J. Li, C. Chen, H. Chen, C. Tong, Towards context-aware social recommendation via individual trust, Knowl.-Based Syst. 127 (2017) 58–66.
[21] M. Ge, F. Gedikli, D. Jannach, Placing high-diversity items in top-n recommendation lists, in: Workshop Chairs, vol. 65, 2011.
[22] S. Vargas, Novelty and diversity enhancement and evaluation in recommender systems and information retrieval, in: Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval, vol. 1281, ACM, 2014.
[23] M. Gan, R. Jiang, Improving accuracy and diversity of personalized recommendation through power law adjustments of user similarities, Decis. Support Syst. 55 (3) (2013) 811–821.
[24] R.H. Li, J.X. Yu, Scalable diversified ranking on large graphs, in: International Conference on Data Mining, 2013, pp. 1152–1157.
[25] M. Kyriakidi, K. Stefanidis, Y. Ioannidis, On achieving diversity in recommender systems, in: Proceedings of the ExploreDB'17, ACM, 2017.
[26] T. Aytekin, Mahmut Özge Karakaya, Clustering-based diversity improvement in top-N recommendation, J. Intell. Inf. Syst. 42 (2014) 1–18.
[27] M. Gan, R. Jiang, ROUND: Walking on an object-user heterogeneous network for personalized recommendations, Expert Syst. 42 (22) (2015) 8791–8804.
[28] L. Yao, Q. Sheng, Y. Qin, X. Wang, S. Ali, Q. He, Context-aware point-of-interest recommendation using tensor factorization with social regularization, in: International ACM SIGIR Conference on Research and Development in Information Retrieval, 2015, pp. 1007–1010.
[29] H. Wu, Z. Zhang, K. Yue, B. Zhang, J. He, L. Sun, Dual-regularized matrix factorization with deep neural networks for recommender systems, Knowl.-Based Syst. 145 (2018) 46–58.
[30] T. Ma, X. Suo, Augmenting matrix factorization technique with the combination of tags and genres, Physica A 461 (2016) 101–116.
[31] W. Wang, G. Zhang, J. Lu, Member contribution-based group recommender system, Decis. Support Syst. 87 (2016) 80–93.
[32] C. Wu, W. Qiu, Z. Zheng, X. Wang, X. Yang, QoS prediction of web services based on two-phase K-means clustering, in: IEEE International Conference on Web Services, 2015, pp. 161–168.

[33] R. Anand, U. Jeffrey David, Mining of Massive Datasets, Cambridge University Press, New York, 2011, pp. 73–126.

[34] Y. Ioannidis, Y. Kotidis, T. Mailis, Ralf Möller, Christian Neuenstadt, L. Özgür, Özçep, Christoforos Svingos, Data mining and query log analysis for scalable temporal and continuous query answering, 2015, http://www.optique-project.eu/.

[35] R. Latha, R. Nadarajan, Analysing exposure diversity in collaborative recommender systems-entropy fusion approach, Physica A (2019).

[36] W. Premchaiswadi, P. Poompuang, et al., Enhancing diversity-accuracy technique on user-based top-n recommendation algorithms, in: IEEE 37th Annual Computer Software and Applications Conference Workshops, 2013.

[37] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, IEEE Comput. 42 (8) (2009) 30–37.