Soft Computing
https://doi.org/10.1007/s00500-022-07416-x

APPLICATION OF SOFT COMPUTING

A hybrid approach to three-way conversational recommendation

Yuan-Yuan Xu[1] · Shen-Ming Gu[2] · Hua-Xiong Li[3] · Fan Min[1]

Accepted: 22 July 2022

Abstract
Conversational recommendation is ubiquitous in e-commerce, while three-way recommendation provides friendly choices for service providers and users. However, their combination has not been studied yet. In this paper, we introduce the three-way conversational recommendation problem and design the hybrid conversational recommendation (HTCR) algorithm to address it. First, a new recommendation problem is defined by considering the man–machine interaction as well as the misclassification and promotion costs. The optimization objective of the problem is to minimize the total cost. Second, a popularity-based technique is designed for user cold-start recommendation, where the user maturity is responsible for deciding when HTCR turns to the second technique. Third, an incremental matrix factorization technique is designed for regular recommendation. It is efficient since only a few rounds of training are needed for newly acquired user feedback. Experiments were carried out on four well-known datasets, including Jester, MovieLens 100K, MovieLens 1M and Yelp. Results demonstrated that our algorithm outperformed state-of-the-art ones in terms of average cost.

Keywords Conversational recommendation · Incremental matrix factorization · Promotion cost · Three-way decision

1 Introduction

Conversational recommendation (McCarthy et al. 2004; Smyth et al. 2004) refers to the scenario that items are recommended to a user based on the feedback. Compared to traditional rating prediction tasks (Sun and Zhang 2018), this scenario focuses on continuous negotiation between the recommender system (RS) and the user. Hence, it is a better model for real-world e-commerce as well as other online RSs. There are some research trends on this topic. To improve the usability, compound critiques (Smyth et al. 2004; Zhang and Pu 2006) help users to understand the suggestions from the RS. By incorporating richer information, sentiment analysis-based dialog (Li et al. 2018) helps in decreasing root mean squared error (RMSE), and chatbot and natural language processing-based approaches (Iovine et al. 2020) provide users with diverse product recommendations. By making use of human–machine interaction, interactive collaborative filtering (CF) with the probabilistic matrix factorization (Zhao et al. 2013) improves the prediction accuracy, and query-based recommendation (Sun and Zhang 2018) helps in decreasing the normalized discounted cumulative gain (NDCG) (Manning et al. 2008) and increasing success rate. However, these work considered only two actions, i.e., "recommend" or "not recommend" items to users.

Three-way recommendation (Zhang and Min 2016) introduces the third action "promote" to the system. This action indicates that RS can consult a user's purchase interest by coupons or discounts. In fact, this action is widely used by most sellers and service providers to increase profit or equivalently, to decrease cost. Naturally, three-way recommendation achieves lower total cost than two-way ones. This problem has also attracted some research interests. The regression-based three-way recommendation (Zhang

✉ Fan Min
minfan@swpu.edu.cn

Yuan-Yuan Xu
yuanyuanxu.cn@gmail.com

Shen-Ming Gu
gushenming666@163.com

Hua-Xiong Li
huaxiongli@nju.edu.cn

1 School of Computer Science, Southwest Petroleum University, Chengdu 610500, People's Republic of China

2 Key Laboratory of Oceanographic Big Data Mining and Application of Zhejiang Province, Zhejiang Ocean University, Zhoushan 316022, People's Republic of China

3 Department of Control and Systems Engineering, School of Management and Engineering, Nanjing University, Nanjing 210093, People's Republic of China

Published online: 11 August 2022

et al. 2017) handles numeric rating data through determining two rating prediction thresholds. Matrix factorization-based approaches (Liu and Ye 2020) obtain better prediction as well as lower total cost. Granular computing-based approaches (Ye and Liu 2021) obtain average cost lower than traditional CF algorithms. Pair-wise learning-based approaches (Huang et al. 2017) improve the performance of RS in terms of weighted accuracy, weighted error and total cost ratio. User reputation-based approaches (Qian et al. 2019) decrease RMSE and mean absolute error (MAE) compared with other popular algorithms. Yet they did not consider the interaction between users and RSs.

In this paper, we introduce the three-way conversational recommendation problem. There are four aspects of the new problem. First, we adopt the new user assumption, so that the whole process of conversation can be tracked. Second, there is a 3×2 cost matrix indicating promotion and misclassification costs. Third, conversation is implemented as follows: During each round, the RS provides a list of recommended and promoted items to the user, and the user provides ratings to some items. This process repeats until the user is not interested in any item in the current list. Fourth, the evaluation measure is the total cost.

Then we design the hybrid conversational recommendation (HTCR) algorithm to address the new problem. It consists of two techniques. The popularity-based technique is designed for user cold-start recommendation, where popularity refers to the times that each item has been rated (Yin et al. 2012; Xu et al. 2017). This is because in the early stage, the user's personal information is so little that any distance-based approach would fail. In order to obtain the lowest cost, we actively learn a pair of popularity thresholds $\left(\pi_\alpha, \pi_\beta\right)$, where $\pi_\alpha > \pi_\beta$. According to the three-way decision (Yao 2010; Li et al. 2016), the thresholds partition the remaining items into three groups: The recommendation candidate pool contains items with popularity not less than π_α, the promotion candidate pool contains items with popularity within (π_β, π_α), and the non-recommendation candidate pool contains all other items. We randomly select L items from first two pools to form the recommendation list, where L is a predefined constant such as 10 or 20 depending on the system settings . Meanwhile, the RS gradually learns more about the user preferences and calculates the user maturity. When the user maturity is no less than a predefined threshold, the regular recommendation technique is triggered.

The incremental matrix factorization (IMF) technique is designed for regular recommendation. At this stage, the RS recommends items to the user according to historical data. Conventional nonnegative matrix factorization algorithms have a high complexity and cannot be incrementally updated when more ratings come (Bucak and Gunsel 2009; Huang et al. 2017). We adopt IMF (Luo et al. 2012; Liu et al. 2020) due to its low time complexity and adaptability to the surge of the amount of new ratings. Some modifications are made to meet our requirements. We also learn a pair of prediction rating thresholds $\left(R_\alpha, R_\beta\right)$ in consideration of cost, where $R_\alpha > R_\beta$. Similar to the popularity-based technique, the candidate item pool are also partitioned into three groups: The recommendation candidate pool contains items with the predicted rating not less than R_α, the promotion candidate pool contains items with the predicted rating within (π_β, π_α), and the non-recommendation candidate pool contains all other items. We also randomly select L items from the first two groups to form the recommendation list.

Experiments were carried out on four datasets including Jester, MovieLens 100K, MovieLens 1M and Yelp. The source code and data are available at https://github.com/FanSmale/TCR. Results demonstrated that the new algorithm was better than other rough set model-based algorithms and popular non-hybrid algorithms in terms of average cost.

The rest of this paper is organized as follows. Section 2 reviews some related work, including recommender systems and the three-way recommendation. Section 3 defines a three-way conversational recommendation problem. Section 4 describes our new algorithm with a running example. Section 5 shows the experimental results. Section 6 concludes and points out some further works.

2 Related work

This section reviews some basic knowledge about recommender systems and three-way recommendation.

2.1 Recommender system

RSs are often used to mine user preferences in e-commerce. There are two main techniques of RSs: content-based and CF. Content-based recommendation adopts the contents of items and establishes the profiles of user preferences for recommendations (Yao et al. 2014). Thus, it is inappropriate for cold-start. CF is a technique to recommend items to customers who share the same interests or experiences. It has different types, including memory-based, model-based and hybrid (Breese et al. 1998; Chen et al. 2014) ones. Memory-based algorithms predict based on the similarity between users or items. In contrast, model-based algorithms use the user database to estimate or learn a model for prediction (Breese et al. 1998). Hybrid algorithms combine at least two approaches to improve the performance (Chen et al. 2014; Paradarami et al. 2017).

Conversational recommendation refers to the scenario that there are multiple rounds of recommendations between users and RSs. Dialog-based approaches (Sun and Zhang 2018; Iovine et al. 2020) formalize it as a dialog scenario and mine users' preferences based on the natural language. Compound critique-based approaches (Smyth et al. 2004; Zhang and Pu 2006) help users to understand the recommendations from RSs. Visualization-based approaches (He et al. 2016; Devendorf et al. 2012) provide users with friendly interfaces. Unfortunately, there is still a lack of formal definition of the scenario.

Nonnegative matrix factorization (NMF) (Lee and Seung 2001) is a popular approach to build CF recommenders. It is also employed in other fields such as face recognition (Meng and Torre 2013) and image processing (Yang et al. 2020). NMF uses the product of two low-rank matrices to approximate the original rating matrix. Some techniques are incorporated into NMF. Sparseness constraint helps in finding part-based representations and converging quickly to the optimal solutions in a few iterations (Hoyer 2004). Bayesian probabilistic models help in avoiding overfitting (Hernando et al. 2016).

Incremental matrix factorization (IMF) (Bucak and Gunsel 2009) is designed to deal with the high complexity of NMF and the incremental feature of online data. Suppose that the rating matrix has a size of $m \times n$, the rank of factorization is k, and the number of iterations for convergence is s. In case of new data arrival, the time complexity of NMF is $O(mnks)$ for matrix updating. In contrast, when the new data of only one user arrive, the time complexity of IMF is $O(nks)$ for matrix updating. Hence, IMF is more appropriate for the conversational scenario of our work.

2.2 Three-way recommendation

Three-way decision (Yao 2010; Liu et al. 2011) induces positive, negative and boundary rules for the decisions of acceptance, rejection and abstaining. It has been introduced in typical classification fields. Sequential three-way decisions for face recognition seek a low decision cost rather than low misclassification error (Li et al. 2016; Zhang et al. 2021). Three-way Email spam filtering provides a more sensible feedback to users for precautionary handling their incoming emails (Zhou et al. 2010). Moreover, three-way recommendations provide a third action "promote" to decrease the average cost (Zhang and Min 2016; Zhang et al. 2017).

Existing three-way recommendation systems (Zhang and Min 2016; Liu and Ye 2020; Xu et al. 2017) include the following four parts: (1) define the user's two preferences: X and $\overline{X}$ means "like" and "dislike," respectively; (2) define a set of actions $A = \{a_P, a_B, a_N\}$ for items, where a_P, a_B and a_N represent "recommend," "promote" and "not recommend," respectively; (3) predict the probability that a user likes an item; and (4) after comparing the probability with the threshold pair (α, β), the action for the item is decided. The optimization objective is to minimize the average cost.

In cost-sensitive learning, cost is a more general evaluation index than accuracy. Existing research proves that incorporating misclassification cost can help in improving the accuracy of testing (Tapkan et al. 2016), and minimizing test cost is considered as the goal of attribute reduction (Min et al. 2014). Existing three-way recommendations (Zhang and Min 2016; Zhang et al. 2017) consider the cost of misclassification and promotion: The former is the penalty for misrecommendation, while the latter is the cost of coupon distribution and discount.

3 Problem statement

In this section, we present the data model, illustrate recommendation scenario and define the problem. Table 1 lists notations used throughout the paper.

3.1 Data model

We consider a recommender system with m users and n items. Let $U = \{u_1, u_2, \dots, u_m\}$ be the set of users and $T = \{t_1, t_2, \dots, t_n\}$ be the set of items. The rating function is given by

$$R : U \times T \to \Omega, \tag{1}$$

where Ω is the rating scale. For convenience, we represent the rating system as an $m \times n$ matrix $R = (r_{ij})_{m \times n}$, where $r_{ij} = R(u_i, t_j)$ for $i \in [1..m]$ and $j \in [1..n]$ is the rating of u_i to t_j. In most applications, R is a sparse matrix with a few valid ratings.

Table 2 lists a rating system where $\Omega = \{1, 2, 3, 4, 5\}$, and 0 means that the user has not browsed or bought the item.

We set the threshold θ to determine whether the user likes the item or not. With this threshold, the user preference matrix is given by $\Delta = (\delta_{ij})_{m \times n}$, where

$$\delta_{ij} = \begin{cases} 1, & \text{if } r_{ij} > \theta; \\ 0, & \text{if } 0 < r_{ij} \le \theta; \\ -1, & \text{otherwise,} \end{cases} \tag{2}$$

where 1, 0 and -1 indicate like, dislike and unknown, respectively.

We also define a list called *gray list*, denoted by $G = \{g_1, g_2, \dots, g_n\}$ for the current user. It records whether an item is still available for recommendation or promotion. Here

$$g_j = \begin{cases} 0, & \text{if } t_j \text{ has been recommended or promoted;} \\ 1, & \text{otherwise.} \end{cases} \tag{3}$$

Table 1 Notations

Notation	Meaning	Comments
$R = (r_{ij})_{m\times n}$	The rating matrix	Data model
$U = \{u_1, \ldots, u_m\}$	The set of users	Data model
$T = \{t_1, \ldots, t_n\}$	The set of items	Data model
θ	The threshold to determine whether a user likes an item or not	Data model
$\varDelta = (\delta_{ij})_{m\times n}$	The matrix of user preference	Data model
A	The set of actions	Data model
$G = \{g_1, \ldots, g_n\}$	The *graylist*	Algorithm variable
$\psi(j)$	The normalized popularity of item t_j	Algorithm variable
$\chi(i)$	The maturity of user u_i	Algorithm variable
q	The number of recommended items	Algorithm variable
$\varGamma$	The recommendation list in each round	Algorithm output
L	The length of the recommendation list	Algorithm setting
$D = (d_{ij})_{m\times n}$	The recommendation matrix	Algorithm output
p_{ij}	The predicted rating for r_{ij}	Algorithm variable
η	The total cost	Algorithm optimization objective

Table 2 Rating matrix R

$U\backslash T$	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}	t_{11}	t_{12}	t_{13}	t_{14}	t_{15}
u_1	5	5	3	0	5	0	0	4	0	3	4	0	0	0	5
u_2	1	4	0	4	0	4	3	0	1	4	1	0	5	3	4
u_3	0	4	0	1	0	0	2	3	0	2	5	0	0	2	1
u_4	1	2	0	0	0	0	0	0	0	0	4	2	5	0	1
u_5	3	0	2	5	0	1	0	0	0	1	0	0	3	3	0
u_6	1	4	3	0	0	0	0	2	1	0	5	4	3	1	4
u_7	0	4	0	1	0	2	4	0	3	0	0	5	3	0	2
u_8	2	1	0	0	5	2	0	1	0	0	4	0	0	0	0
u_9	0	5	4	0	0	0	4	0	0	0	4	4	0	5	4
u_{10}	0	5	5	5	0	0	0	5	0	3	3	0	2	0	0

Table 3 Cost matrix C

Action	Like	Dislike
a_P	λ_{PP}	λ_{PN}
a_B	λ_{BP}	λ_{BN}
a_N	λ_{NP}	λ_{NN}

In the three-way recommendation scenario, the recommendation matrix is given by $D = (d_{ij})_{m\times n}$, and

$$d_{ij} = \begin{cases} a_P, & \text{if } p_{ij} > R_\alpha; \\ a_N, & \text{if } 0 < p_{ij} < R_\beta; \\ a_B, & \text{otherwise}, \end{cases} \tag{4}$$

where a_P, a_N and a_B indicate "recommend," "not recommend" and "promote." R_α and R_β are ratings thresholds mapped from the probability threshold α and β. p_{ij} is the predicted rating for r_{ij}.

Table 3 lists the cost matrix as the foundation of three-way decisions.

With the recommendation matrix D and the like threshold θ, the set of user item pairs with known ratings is partitioned into six subsets:

$$\begin{aligned} PP &= \{(i,j) \in [1..m] \times [1..n] | r_{ij} \in \varOmega \wedge d_{ij} = a_P \wedge r_{ij} > \theta\}, \\ PN &= \{(i,j) \in [1..m] \times [1..n] | r_{ij} \in \varOmega \wedge d_{ij} = a_P \wedge r_{ij} \leq \theta\}, \\ BP &= \{(i,j) \in [1..m] \times [1..n] | r_{ij} \in \varOmega \wedge d_{ij} = a_B \wedge r_{ij} > \theta\}, \\ BN &= \{(i,j) \in [1..m] \times [1..n] | r_{ij} \in \varOmega \wedge d_{ij} = a_B \wedge r_{ij} \leq \theta\}, \\ NP &= \{(i,j) \in [1..m] \times [1..n] | r_{ij} \in \varOmega \wedge d_{ij} = a_N \wedge r_{ij} > \theta\}, \\ NN &= \{(i,j) \in [1..m] \times [1..n] | r_{ij} \in \varOmega \wedge d_{ij} = a_N \wedge r_{ij} \leq \theta\}. \end{aligned} \tag{5}$$

Consequently, the total cost is given by

$$\begin{aligned} \eta = {} & \lambda_{PP}|PP| + \lambda_{PN}|PN| + \lambda_{BP}|BP| \\ & + \lambda_{BN}|BN| + \lambda_{NP}|NP| + \lambda_{NN}|NN|, \end{aligned} \tag{6}$$

where $|\cdot|$ denotes the cardinality of a set.

3.2 The conversational recommendation scenario

Figure 1 illustrates the conversational recommendation scenario. The left side of the dotted line is the behavior of a user, and the right side is the behavior of the RS. These behaviors are explained in detail as follows.

① A user enters the RS through logging in a website or an App.

② The RS predicts the user's ratings to items.
③ The RS recommend a list of items to the user.
④ The user evaluates the recommended items.
⑤ If no recommended item favors the user, she will quit the system.
⑥ If the user finds some favorite items in the list, she buys them.
⑦ The user provides the feedback by numeric rating.
⑧ The recommender records the choices of the user and updates *gray list* (mentioned in Section 3.1).
⑨The RS enters the next round of recommendation.

3.3 The problem

We also consider the leave-user-out training–testing scenario. That is, during the period that a new user enters and exists the system, the information of other users remains unchanged. Naturally, to obtain good recommendation results for all users (i.e., min η), we only need to consider the optimization for each user. For user u_i, we have

$$\begin{aligned} PP_i &= \{(i,j) | j \in [1..n] \wedge r_{ij} \in \Omega \wedge d_{ij} = a_P \wedge r_{ij} > \theta\},\\ PN_i &= \{(i,j) | j \in [1..n] \wedge r_{ij} \in \Omega \wedge d_{ij} = a_P \wedge r_{ij} \le \theta\},\\ BP_i &= \{(i,j) \in [1..m] \times [1..n] | r_{ij} \in \Omega \wedge d_{ij} = a_B \wedge r_{ij} > \theta\},\\ BN_i &= \{(i,j) \in [1..m] \times [1..n] | r_{ij} \in \Omega \wedge d_{ij} = a_B \wedge r_{ij} \le \theta\},\\ NP_i &= \{(i,j) \in [1..m] \times [1..n] | r_{ij} \in \Omega \wedge d_{ij} = a_N \wedge r_{ij} > \theta\},\\ NN_i &= \{(i,j) \in [1..m] \times [1..n] | r_{ij} \in \Omega \wedge d_{ij} = a_N \wedge r_{ij} \le \theta\}. \end{aligned} \tag{7}$$

Consequently, the cost of u_i is given by

$$\begin{aligned} \eta_i = &\lambda_{PP}|PP_i| + \lambda_{PN}|PN_i| + \lambda_{BP}|BP_i| \\ &+ \lambda_{BN}|BN_i| + \lambda_{NP}|NP_i| + \lambda_{NN}|NN_i|. \end{aligned} \tag{8}$$

Now we can define the new problem as follows.

Problem 1 Three-way conversational recommendation
Input: A rating system R, a cost matrix C, the current user u_i.
Output: The recommendation lists $\Gamma_1, \ldots, \Gamma_k$ where k is the round of interactions.
Optimization objective: min η_i.

Here we have a few lists $\Gamma_1, \ldots, \Gamma_k$ for different rounds. Suppose that Algorithm A produces a better Γ_1 than Algorithm B, while Algorithm B produces a better Γ_2 than Algorithm A, which algorithm is better? The answer is uncertain. In other words, local optima does not assure global optima for the user.

4 The algorithm

In this section, we first describe the algorithm and two recommendation techniques, then present a running example.

According to Problem 1, the number of rounds s indicates the adherence of the user to the system. With a larger s, more items have the opportunity of being recommended; hence, $|NP|$ is smaller and $|PP|$ is bigger, which decrease the total cost. At the same time, $|PN|$ is bigger, which increases the total cost. Hence, we need to adjust the recommendation list to achieve a tradeoff among different costs.

Recommender systems usually have two tasks: exploiting and exploring. Exploiting refers to finding items that favor the user, while exploring refers to finding items that potentially favor the user. To balance them, we divide each recommendation list into two parts: one for high-confidence items for the exploiting task and the other for medium-confidence items for the exploring task.

4.1 Algorithm description

Algorithm 1 presents the one-round recommendation under the conversational scenario. Specifically, the input ξ is the maturity threshold deciding the recommendation technique to employ, and the output Γ is the recommendation list.

Line 1 initializes both the recommendation (RC) and promotion (PC) candidate sets to be empty.

Line 2 decides which technique to use.

Lines 3–9 handle cold-start recommendation with the popularity-based approach. Popularity refers to the times that an item has been rated. A more popular item often favors more people. In the previous work (Xu et al. 2017), we have found that recommending some items with lower popularity can decrease costs. So we actively learn a pair of popularity thresholds (π_α, π_β), where $\pi_\alpha > \pi_\beta$, to divide candidate items into three groups: items with popularity no less than π_α form the recommendation candidate set; items with popularity within the interval (π_β, π_α) form the promotion candidate set; and other items form the remaining set.

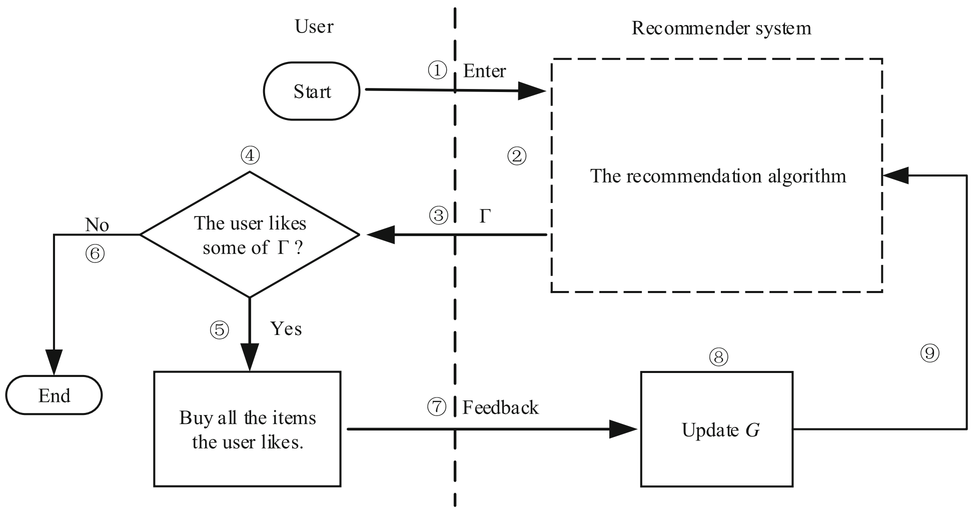
Line 10 updates the user's maturity $\chi(i)$. We will discuss this issue in more detail in Sect. 4.1.1.

Lines 12–21 handle regular recommendation with an IMF-based approach. We also learn a threshold pair $\left(R_\alpha, R_\beta\right)$ to partition candidate items into three groups, while the semantics of the thresholds is changed to accommodate rating prediction.

Finally, Line 23 randomly selects q and $(L-q)$ items from the recommendation and promotion candidate sets, respectively.

Figure 2 describes the recommendation process from another perspective. Figure 2a shows that the maturity is responsible for switching the recommendation techniques.

Fig. 1 Conversational recommendation scenario


User
Recommender system
Start
① Enter
The recommendation algorithm
②
③ Γ
④
The user likes some of Γ ?
No
⑥
⑤ Yes
End
Buy all the items the user likes.
⑦ Feedback
⑧
Update G
⑨

Algorithm 1 Hybrid conversational recommendation (HTCR) algorithm.

```
Input: $u_i, T, R, G, R_\alpha, R_\beta, \xi, m, n, \pi_\alpha, \pi_\beta$
Output: Recommendation list $\Gamma$

1: $RC = \varnothing$, $PC = \varnothing$; // Candidate recommendation/promotion list
2: if $(\chi(i) < \xi)$ then
3:    for $(t_j \in T)$ do
4:       if $(g_j = 1$ and $\psi(j) \geq \pi_\alpha)$ then
5:          $RC = RC \cup t_j$;
6:       else if $(g_j = 1$ and $\psi(j) \in (\pi_\beta, \pi_\alpha))$ then
7:          $PC = PC \cup t_j$;
8:       end if
9:    end for
10:   update $\chi(i)$;
11: else
12:   for $(t_j \in T)$ do
13:      if $(g_j = 1)$ then
14:         Predict $P_i$-based IMF; //$P_i$ is the predicted ratings for $u_i$ in
            Algorithm 2;
15:         if $(p_{ij} \geq R_\alpha)$ then
16:            $RC = RC \cup t_j$;
17:         else if $(p_{ij} \in (R_\beta, R_\alpha))$ then
18:            $PC = PC \cup t_j$;
19:         end if
20:      end if
21:   end for
22: end if
23: $\Gamma = q$ items randomly selected from $RC \cup (L - q)$ items randomly
    selected from $PC$;
24: return $\Gamma$;
```

The two threshold pairs (π_α, π_β) or (R_α, R_β) are considered as the input of each algorithm. Figure 2b–c show that the candidate items are divided into three parts: recommendation candidates, promotion candidates and non-recommendation candidates. Finally, Fig. 2d shows that q and $(L - q)$ items are selected randomly from RC and PC to form Γ.

4.1.1 The popularity-based recommendation technique

In the first stage, the popularity-based recommendation is employed to deal with the cold-start problem. We use user maturity to switch between the two techniques. Equation 9 indicates the maturity matrix

$$Z = \begin{bmatrix} z_{00} & z_{01} \\ z_{10} & z_{11} \end{bmatrix}. \tag{9}$$

According to He et al. (2016), both existing and nonexisting ratings are beneficial for RS to understand users. In case that a user has not rated an item with high popularity, it is likely that the user knows the item but is unwilling to rate it. So it can be inferred that she does not like it. Therefore, items without ratings also have implicit feedback information, so it is considered as follows: (1) for each recommendation to existing rating, the maturity increases by z_{00}; (2) for each recommendation to nonexisting rating, the maturity increases by z_{01}; (3) for each promotion to existing rating, the maturity increases by z_{10}; and (4) for each promotion to nonexisting rating, the maturity increases by z_{11}.

So the maturity of u_i is given by

$$\begin{aligned}\chi(i) = {} & |(i,j)|t_j \in RC \wedge r_{ij} \neq 0| \times z_{00} \\ & + |(i,j)|t_j \in RC \wedge r_{ij} = 0| \times z_{01} \\ & + |(i,j)|t_j \in PC \wedge r_{ij} \neq 0| \times z_{10} \\ & + |(i,j)|t_j \in PC \wedge r_{ij} = 0| \times z_{11}. \end{aligned} \tag{10}$$

If $\chi(i) > \xi$, we choose the IMF-based technique to make recommendations.

The steps of popularity-based recommendations are as follows. First, we define the popularity as the number of times each item has been rated. Then, we use the maximum popularity to normalize the popularity of each item. The function $\psi(j)$ represents the normalized popularity of item t_j. Third,

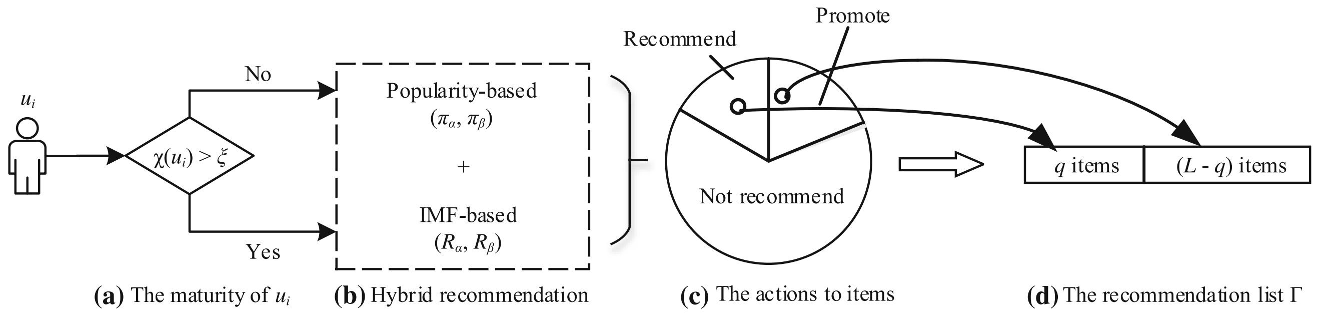


Fig. 2 Three-way hybrid recommendation

we obtain a pair of popularity threshold (π_α, π_β) by active learning, where $\pi_\beta < \pi_\alpha$. If $\psi(j) \geq \pi_\alpha$, t_j enters the set of recommendation candidates. If $\psi(j) \in (\pi_\beta, \pi_\alpha)$, t_j enters the set of promotion candidates. Otherwise, t_j can be ignored.

4.1.2 The IMF-based recommendation technique

In the second stage, IMF is employed for regular recommendations. In this stage, the user is not new and there are more new rating coming into our datasets. IMF is more efficient than conventional NMF for handling incremental data. The time complexity of IMF is lower than conventional NMF for handling incremental data.

NMF (Lee and Seung 2001) approximately factorizes R into two factor matrices $V (V \in \mathbb{R}^{m\times k})$ and $H (H \in \mathbb{R}^{k\times n})$

$$R \approx VH, \tag{11}$$

where k is the rank of factorization. V and H denote that users and items are mapped into a k-dimensional latent space. So, we call V as user subspace and H as item subspace. Generally, the NMF technique minimize loss function F to optimize the results of factorization

$$\begin{aligned}\arg\min_{V,H} F &= \|R - VH\|_F^2 \\ &= \sum_{i=1}^{m}\sum_{j=1}^{n}(R_{ij} - \sum_{l=1}^{k} V_{il}H_{lj})^2,\end{aligned} \tag{12}$$

where $\|\cdot\|_F$ means Frobenius norm. It has a high complexity of $O(mnk)$ in each iteration. Meanwhile, it does not work for online datasets and incremental data.

So we consider IMF and make some modifications to suit our needs. We review the optimal objective function of the regularized MF (Guan et al. 2011)

$$\begin{aligned}\arg\min_{V,H} F &= \|W \odot (R - VH)\|_F^2 \\ &\quad + \varepsilon_1 \sum_{i=1}^{m} \|V_i\|_2^2 + \varepsilon_1 \sum_{j=1}^{n} \|H_j\|_2^2,\end{aligned} \tag{13}$$

where $\odot$ means inner product and $\|\cdot\|$ means L_2 norm. $W = (\omega_{ij})_{m\times n}$ records the existing entries of R. If r_{ij} exists, then $\omega_{ij} = 1$, otherwise 0. ε_1 controls the strength of regularization, which are usually an L_2 norm to prevent overfitting.

Algorithm 2 describes our IMF algorithm for u_i. The input V and H are two subspaces obtained by NMF on the original dataset R. k is the rank of subspaces. e is the convergency threshold. ε_2 is the learning rate. The output V' and H' are the new subspaces updated by IMF.

Line 1 employs the input V and H as initial values in the first iteration.

Line 2 assigns the maximum iteration times as a constant N.

Line 3 computes the predictive value P_i by $V_i^{(s-1)}$ and $H^{(s-1)}$. The subscript i indicates the ith row of the matrix.

Lines 4–7 exclude the nonexisting ratings.

Line 8 computes the residual.

Lines 9–12 update two subspaces. Line 10 updates V by the values of V and H from last iteration. Line 11 updates H by the value of V from Line 10 and the value of H from the last iteration.

Lines 13–15 judge whether the iteration converges. Line 13 computes 1-norm of the difference between $R_i^{(s)}$ and $P_i^{(s-1)}$. Line 14 expresses that if the norm is below e, the iteration is terminated.

Lines 18–19 return the values of V and H after the end of the iteration.

According to the three-way decision, there are a pair of probability threshold (α, β) to decide d_{ij}. However, we use popularity and prediction rating to decide d_{ij}, respectively, in two stages, so we design a function Eqs. (14)–(16) to map probability to popularity and rating,

$$(\pi_\alpha^*, \pi_\beta^*) = \arg\min_{0<\pi_\beta<\pi_\alpha\leq 1} \eta(\pi_\alpha, \pi_\beta), \tag{14}$$

$$R_\alpha^* = \arg\min_{R_\alpha} \eta(\theta, R_\alpha), \text{ if } R_\alpha > \theta, \tag{15}$$

$$R_\beta^* = \arg\min_{R_\beta,\theta} \eta(R_\beta, \theta), \text{ if } R_\beta < \theta. \tag{16}$$

$(\pi_\alpha^*, \pi_\beta^*)$ and (R_α^*, R_β^*) are the optimal solutions.

Algorithm 2 Incremental matrix factorization (IMF) algorithm.

```
Input: $R, u_i, k, V, H, \varepsilon_1, \varepsilon_2, e$
Output: $P_i$

1: $V^{(0)} = V, H^{(0)} = H$;
2: for ($s$ = 1 to $N$) do
3:   $P_i = V_i^{(s-1)} H^{(s-1)}$;
4:   for ($j$ = 1 to $n$) do
5:     if ($r_{ij} = 0$) then
6:       continue; // nonexisting rating
7:     end if
8:     $res = r_{ij} - p_{ij}$; // Residual
9:     for ($l$ = 1 to $k$) do
10:      $v_{il}^{(s)} = v_{il}^{(s-1)} + \varepsilon_2 * \left(2 * res * h_{lj}^{(s-1)} - \varepsilon_1 * v_{il}^{(s-1)}\right)$; // Update $V$
11:      $h_{lj}^{(s)} = h_{lj}^{(s-1)} + \varepsilon_2 * \left(2 * res * v_{il}^{(s)} - \varepsilon_1 * h_{lj}^{(s-1)}\right)$; // Update $H$
12:    end for
13:    if $\left(\|R_i^{\prime(s)} - P_i^{(s-1)}\|_1 < e\right)$ then
14:      break; // Converge
15:    end if
16:  end for
17: end for
18: return $P_i$;
```

4.2 Time complexity analysis

The time complexity of our HTCR algorithm is the sum of the popularity-based and the IMF-based recommendation algorithms. We will elaborate it as follows.

4.2.1 The time complexity of the popularity-based recommendation

Proposition 1 *The time complexity of the popularity-based recommendation is $O(n)$.*

Proof Lines 3–11 in Algorithm 1 describe the popularity-based recommendation. The **for** loop of Lines 3–9 takes $O(n)$ of time. Line 11 updates the maturity of u_i by Eq. (10) with $O(n)$ of time. Line 23 computes the recommendation list with $O(n)$ of time. Hence, one round of popularity-based recommendation takes $O(n) + O(n) + O(n) = O(n)$ of time. This completes the roof. □

If we recommend M rounds, the time complexity is $O(Mn)$. The upper bound of M is $\lfloor n/L \rfloor$. Due to the maturity threshold ξ, M is usually a small number in reality. It is hardly influenced by n. So we will not replace it with n/L in the following analysis.

4.2.2 The time complexity of the IMF-based recommendation

Proposition 2 *The time complexity of the IMF-based recommendation is $O(Nnk)$.*

Proof Lines 12–22 in Algorithm 1 describe the IMF-based recommendation. Line 12 makes n loops with $O(n)$ of time. Line 14 predicts ratings for u_i by Algorithm 2. Table 4 describes the detailed complexity of Algorithm 2, and we will analyze it as follows.

In Algorithm 2, Line 3 obtains the product of V_i and H. Since the scan is relevant to n and k, the time complexity is $O(nk)$. Lines 5–7 judge whether the current rating exists; hence, the time complexity is $O(1)$. Lines 9–12 update the i^{th} row of V and the j^{th} column of H, and the time complexity is $O(k)$. Lines 13–15 scan each entry of R_i' in two iterations to compute 1-norm; hence, the time complexity is $O(k)$. Suppose it takes N rounds to converge and the total time complexity of Algorithm 2 is

$$O(N) * (O(nk) + O(1) + O(k) + O(mk)) = O(Nnk) \tag{17}$$

Lines 15–19 in Algorithm 1 take $O(1)$ of time. Line 23 computes the recommendation list with $O(n)$ of time. Hence, one round of IMF-based recommendation takes $O(n) + O(Nnk) + O(1) + O(n) = O(Nnk)$ of time. □

4.2.3 Overall time complexity of the HTCR algorithm

The performance of the HTCR algorithm should be evaluated across the whole dataset. Hence, we need the complete leave-user-out training–testing process. That is, each time we select one user as the testing data and all the others as the training data. In this way, the recommendation process will repeat m times. We have the following corollary.

Corollary 1 *In a complete leave-user-out training–testing process, the overall time complexity of the HTCR algorithm is $O(mn(M + Nk))$.*

Proof According to Proposition 1, the cold-start recommendation process takes $O(nM)$ of time for each user. According to Proposition 2, the IMF recommendation process takes $O(nNk)$ of time for each user. Hence, the time complexity for all users is $O(m) * O(nM + nNk) = O(mn(M + Nk))$. □

4.3 A running example

Figure 3 shows the entire recommendation process of u_2 of Table 2, including three rounds of conversational recommendation. The main parameters are as follows. π_α, π_β, R_α, R_β, ξ and Z are 0.7, 0.3, 4, 2.5 and $\begin{bmatrix} 5 & 2.5 \\ 4 & 2 \end{bmatrix}$, respectively.

Round I adopts the popularity-based recommendation. The popularity of t_1–t_{15} is 0.67, 1.00, 0.56, 0.56, 0.22, 0.44, 0.44, 0.56, 0.33, 0.56, 0.89, 0.44, 0.67, 0.56 and 0.78, respectively. Naturally, we obtain $\psi(i) > \pi_\alpha$ for $i \in \{2, 11, 5\}$ and

Table 4 Time complexity of Algorithm 2

Lines	Complexity
Line 3	$O(nk)$
Lines 5–7	$O(1)$
Lines 9–12	$O(k)$
Lines 13–15	$O(k)$
Total	$O(N) * (O(nk) + O(1) + O(k) + O(k)) = O(Nnk)$

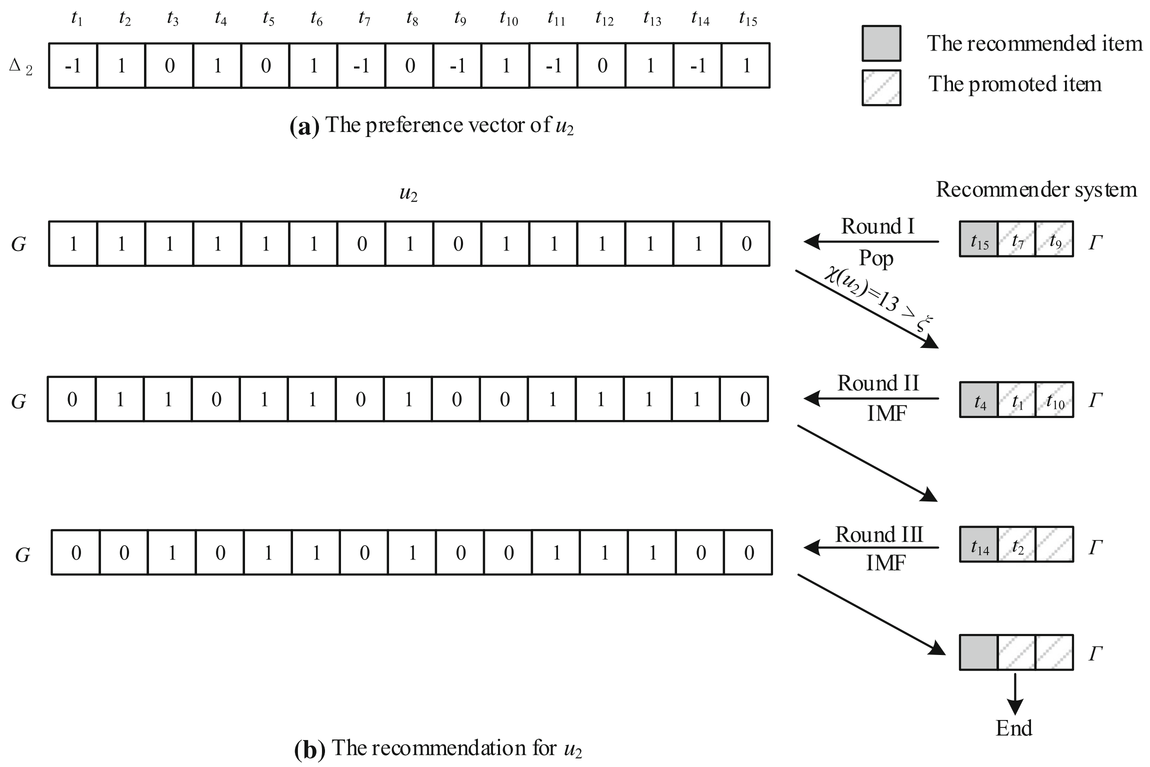

t1 t2 t3 t4 t5 t6 t7 t8 t9 t10 t11 t12 t13 t14 t15
Δ2 -1 1 0 1 0 1 -1 0 -1 1 -1 0 1 -1 1
The recommended item
The promoted item
(a) The preference vector of u2
u2
Recommender system
G 1 1 1 1 1 1 0 1 0 1 1 1 1 1 0
Round I
Pop
t15 t7 t9 Γ
χ(u2)=13 > ξ
G 0 1 1 0 1 1 0 1 0 0 1 1 1 1 0
Round II
IMF
t4 t1 t10 Γ
G 0 0 1 0 1 1 0 1 0 0 1 1 1 0 0
Round III
IMF
t14 t2 Γ
Γ
End
(b) The recommendation for u2

Fig. 3 Running example

$\pi_\beta < \psi(i) < \pi_\alpha$ for $i \in \{1, 3\text{–}10, 12\text{–}14\}$. So, $RC = \{2, 11, 15\}$ and $PC = \{1, 3\text{–}10, 12\text{–}14\}$. t_{15} is randomly selected for recommendation; t_7 and t_9 are randomly selected for promotion; that is, $\Gamma = (15, 7, 9)$. Figure 3a shows that $\Delta_2 = (0, 1, -1, 1, -1, 1, 0, -1, 0, 1, 0, -1, 1, 0, 1)$, so t_{14} is successfully recommended in this round. In this round, $\chi(2) = 13 (> 5)$ and she is matured. So we enter the regular recommendation.

Round II adopts the IMF-based recommendation. $p_{2,4}$ and $p_{2,13}$ are greater than R_α, while $p_{2,1}$, $p_{2,6}$, $p_{2,10}$ and $p_{2,11}$ are within (R_β, R_α). So, $RC = \{4, 13\}$, $PC = \{1, 6, 10, 11\}$, and $\Gamma = (4, 1, 10)$. Finally, t_4 and t_{10} are successfully recommended and promoted, respectively.

Round III continues to use the IMF-based recommendation. In this round, $RC = \{14\}$, $PC = \{2\}$ and $\Gamma = (14, 2)$. Only t_2 is successfully promoted. At the end, we cannot find any more items for u_2 and the recommendation is terminated. Hence, the total cost for u_2 is 120.

5 Experiments

In this section, we present the experimental results to answer these following questions.

(1) Is HTCR efficient?
(2) Is HTCR better than non-hybrid ones?
(3) Is the three-way decision model superior to other rough set models?

5.1 Datasets and experiment settings

2. The paper can also compare with one more dataset and provide the details. detail: the popularity threshold pair π_α, π_β, the rating threshold pair R_α, $R_\beta$the maturity threshold ξ.

Table 5 shows basic information of three datasets Jester, MovieLens 100K, MovieLens 1M and Yelp. Only user IDs, item IDs and ratings were considered in our experiments.

Table 5 Description of experimental datasets

Dateset	Users	Items	Ratings
Jester	24,983	100	1,810,455
MovieLens 100K	943	1682	100,000
MovieLens 1M	6040	3952	1,000,209
Yelp	11,916	3815	133,958

For example, in MovieLens 100K, there were 1682 movies in total, 563 were rated less than 10 times, 526 were rated 10–50 times, 259 were rated 50–100 times, 302 movies were rated 100–300 times, and only 32 movies were rated more than 300 times. Consequently, 80% of the films were rated less than 100 times. After normalized popularity, π_α and π_β we learned were 0.4 and 0.1, respectively.

We employed the following two cost matrices for experimentation:

$$C_1 = \begin{bmatrix} 0 & 40 \\ 10 & 20 \\ 40 & 0 \end{bmatrix} \quad \text{and} \quad C_2 = \begin{bmatrix} 0 & 50 \\ 20 & 20 \\ 30 & 0 \end{bmatrix}.$$

5.2 The efficiency of the HTCR algorithm

Figure 4 shows the runtime change of our HTCR algorithm with different numbers of users or items on the MovieLens 1M dataset. The runtime increased linearly wrt the number of users or items.

Figure 4b shows that when the number of users changes from 500 to 6040, the runtime increases from 1.5095 to 14.6970 sec. For example, when the number of users was 6040, the total runtime was 14.6970 sec. It took about 5.6723 sec to perform overhead operations such as data loading and space allocation. We also noticed that when the number of users increased by 500, the runtime increased by 1.5 sec on average. Figure 4(b) shows that when the number of items changes from 400 to 3952, the runtime increases from 1.5611 to 14.6940 sec. When the number of items increased by 400, the runtime increased by 1.2 sec on average.

5.3 Comparison with non-hybrid algorithms

Table 6 shows the average costs of HTCR on three datasets are less than those of non-hybrid algorithms. “Pop” and “IMF” mean that we only employ the popularity-based or the IMF-based recommendation, respectively. Table 6a shows that when inputting C_1 on Jester, the average cost of HTCR was 11.5235% less than Pop and 4.9280% less than IMF. Table 6b shows that when inputting C_2 on Jester, the average cost of HTCR was 3.5380% less than Pop and 13.6245% less than IMF. So, the superiority of our HTCR algorithm is obvious.

5.4 Comparison with other rough set models

Tables 7 and 8 show the experimental results of Pawlak, VPRS and three-way models. Table 7 demonstrates that average costs of the three-way model on three datasets are less than those of Pawlak model. Table 7a, b shows that on MovieLens 1M, the average cost of three-way model is 23.7030% or 22.3197% less than that of Pawlak, respectively. So, the superiority of three-way model is still obvious.

For simplicity, Table 8 shows only the average costs of MovieLens 1M under VPRS model. Both R_α and R_β have a step size of 0.5, and R_α is greater than R_β. The entire change is presented in the form of a lower left triangular matrix. It can be seen that the lowest average cost occurs at (2.5, 3.5). Both of the values are nearly the pair of thresholds (2.8, 3.8) calculated from the three-way decision. This proves that our model is more efficient than the VPRS model to minimize the average cost.

5.5 Discussions

Now we can answer the question at the beginning of this section.

(1) Our HTCR algorithm is efficient. It is linear wrt the number of users or items, hence scalable to large datasets. In the parallel environment, each user can be recommended independently; therefore, the time complexity is $O(n(M + Nk))$, which is not influence by the number of users. In each round of recommendation, the time complexity is $O(nk)$, which can be finished in a few milliseconds. In other words, our algorithm is appropriate for real-time recommendation.
(2) Our HTCR algorithm performs better than the non-hybrid algorithms in term of average cost. The cost decrease were 3.4906–25.6494%.
(3) Three-way model performs better than Pawlak and VPRS models. Compared with Pawlak model, our cost decrease were 13.6765–31.4850%. Compared with VPRS model, our model can find the thresholds more efficiently.

We also found that our HTCR algorithm performed stably, and the cost matrix setting did not influence its superiority.

6 Conclusions and future work

In this paper, we proposed a hybrid approach to three-way conversational recommendation. The results showed that our algorithm was superior to other existing models. The following research topics deserve further investigation:

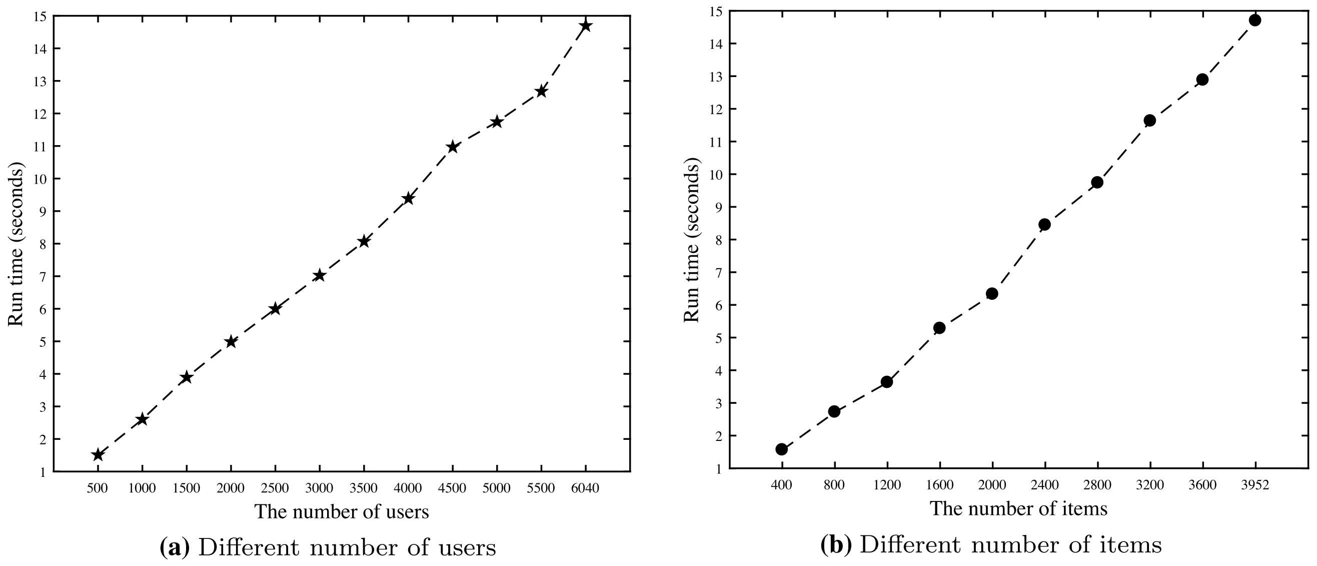


Fig. 4 Runtime of HTCR on MovieLens 1M dataset

Table 6 Comparisons of HTCR and non-hybrid

Dateset	Pop	IMF	HTCR
(a) With cost matrix C_1			
Jester	20.8539 ± 0.0068	19.4886 ± 0.0156	**18.5282 ± 0.0013**
MovieLens 100 K	20.3154 ± 0.0199	17.5905 ± 0.0618	**16.2129 ± 0.1961**
MovieLens 1 M	21.3876 ± 0.0063	17.2543 ± 0.1870	**16.2534 ± 0.0101**
Yelp	21.7557 ± 0.0063	24.1562 ± 0.0260	**18.9837 ± 0.0278**
(b) With cost matrix C_2			
Jester	27.0483 ± 0.0065	24.2200 ± 0.0102	**23.3631 ± 0.0028**
MovieLens 100 K	26.5448 ± 0.0258	21.4247 ± 0.0285	**20.2157 ± 0.2190**
MovieLens 1 M	26.8660 ± 0.0034	21.3146 ± 0.0625	**20.5706 ± 0.2360**
Yelp	26.2255 ± 0.0116	28.8899 ± 0.0345	**21.4798 ± 0.0319**

We repeated experiments ten times on each dataset and expressed the average costs in the form of "mean value ± standard deviation." Bold fonts indicate the lowest cost of each row

Table 7 Comparisons of HTCR and Pawlak

Dateset	Pawlak	HTCR	Improvement (%)
(a) With cost matrix C_1			
Jester	21.9455 ± 0.0008	**18.5282 ± 0.0013**	↓ 15.5718
MovieLens 100 K	21.3522 ± 0.0260	**16.2129 ± 0.1961**	↓ 23.8734
MovieLens 1 M	21.3028 ± 0.0027	**16.2534 ± 0.0101**	↓ 23.7030
Yelp	25.5239 ± 0.0023	**18.9827 ± 0.0278**	↓ 25.6277
(b) With cost matrix C_2			
Jester	27.0646 ± 0.0077	**23.3631 ± 0.0028**	↓ 13.6765
MovieLens 100 K	26.4744 ± 0.0519	**20.2157 ± 0.2190**	↓ 23.6406
MovieLens 1 M	26.4811 ± 0.0373	**20.5706 ± 0.2360**	↓ 22.3197
Yelp	31.3505 ± 0.0005	**21.4798 ± 0.0319**	↓ 31.4850

Bold fonts indicate the lowest cost of each row. We repeated experiments ten times on each dataset and expressed the average costs in the form of "mean value ± standard deviation"

Table 8 Average cost of VPRS when inputting C_1 on MovieLens 1M

$R_\alpha \backslash R_\beta$	1.0	1.5	2.0	2.5	3.0	3.5	4.0
1.5	17.2379						
2.0	16.9738	16.8509					
2.5	16.7195	16.5918	16.5068				
3.0	16.5226	16.3984	16.3134	16.4125			
3.5	16.4301	16.3047	16.2888	**16.2697**	16.7258		
4.0	16.5013	16.3759	16.2909	16.3900	16.7970	17.7531	
4.5	16.7940	16.6686	17.5836	17.6827	18.0897	19.0458	20.5642

The bold font indicates the lowest cost

(1) Better cold-start techniques. This work only considers the popularity-based cold-start recommendation. More sophisticated techniques could be applied or designed for this issue.
(2) Richer data sources. This work is limited to the basic rating matrix. We will design new algorithms to process demographic information, item features, reviews and social network. In this way, the recommendation performance can be enhanced.
(3) Dynamic updating of the information. This work only considers the conversational recommendation of each user. In reality, many users update their rating information simultaneously. We should design online data updating and off-line subspace updating strategies.

Funding This work is supported in part by the National Natural Scientific Foundation of China (71671086, 61976194, 41631179), the Open Project of Key Laboratory of Oceanographic Big Data Mining and Application of Zhejiang Province (OBDMA202005), the Zhejiang Provincial Natural Science Foundation of China (LY18F-030017), the Natural Science Foundation of Sichuan Province (2019YJ0314), the Scientific Innovation Group for Youths of Sichuan Province (2019JDTD0017), the Central Government Funds of Guiding Local Scientific and Technological Development (2021ZYD0003), the Applied Basic Research Project of the Science and Technology Bureau of Nanchong City (SXHZ040).

Data availability The data are publicly available online. The datasets generated during and/or analyzed during the current study are available in the Github repository, https://github.com/FanSmale/TCR.

Declarations

Conflict of interest The authors have no conflicts of interest to declare that are relevant to the content of this article.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

Breese JS, Heckerman D, Kadie C (1998) Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the 14th conference on uncertainty in artificial intelligence. Morgan Kaufmann Publishers Inc., pp 43–52

Bucak SS, Gunsel B (2009) Incremental subspace learning via non-negative matrix factorization. Pattern Recogn 42(5):788–797

Chen W, Niu ZD, Zhao XY, Li Y (2014) A hybrid recommendation algorithm adapted in e-learning environments. World Wide Web 17(2):271–284

Devendorf L, O'Donovan J, Höllerer T (2012) Topiclens: an interactive recommender system based on topical and social connections. In: Proceedings of the 1st international workshop on recommendation technologies for lifestyle change, pp 41–47

Guan N, Tao D, Luo Z, Yuan B (2011) Manifold regularized discriminative nonnegative matrix factorization with fast gradient descent. IEEE Trans Image Process 20(7):2030–2048

He XN, Zhang HW, Kan MY, Chua TS (2016) Fast matrix factorization for online recommendation with implicit feedback. In: SIGIR, pp 549–558

He C, Parra D, Verbert K (2016) Interactive recommender systems: A survey of the state of the art and future research challenges and opportunities. Expert Syst Appl 56:9–27

Hernando A, Bobadilla J, Ortega F (2016) A non negative matrix factorization for collaborative filtering recommender systems based on a Bayesian probabilistic model. Knowl-Based Syst 97:188–202

Hoyer PO (2004) Non-negative matrix factorization with sparseness constraints. J Mach Learn Res 5:1457–1469

Huang JJ, Wang J, Yao YY, Zhong N (2017) Cost-sensitive three-way recommendations by learning pair-wise preferences. Int J Approx Reason 86:28–40

Huang XP, Wu L, Chen EH, Zhu HS, Liu Q, Wang YJ (2017) Incremental matrix factorization: a linear feature transformation perspective. IJCA I:1901–1908

Iovine A, Narducci F, Semeraro G (2020) Conversational recommender systems and natural language: a study through the converse framework. Decis Support Syst 131:113250

Lee DD, Seung HS (2001) Algorithms for non-negative matrix factorization. In: NIPS, pp 556–562

Li HX, Zhang LB, Huang B, Zhou XZ (2016) Sequential three-way decision and granulation for cost-sensitive face recognition. Knowl-Based Syst 91:241–251

Li R, Kahou SE, Schulz H, Michalski V, Charlin L, Pal C (2018) Towards deep conversational recommendations. In: NIPS, pp 9725–9735

Liu D, Ye XQ (2020) A matrix factorization based dynamic granularity recommendation with three-way decisions. Knowl-Based Syst 191:105243

Liu D, Li TR, Ruan D (2011) Probabilistic model criteria with decision-theoretic rough sets. Inf Sci 181(17):3709–3722

Liu WQ, Luo LK, Peng H, Zhang LM, Wen W, Wu H, Shao W (2020) A three-stage method for batch-based incremental nonnegative matrix factorization. Neurocomputing 400:150–160

Luo X, Xia YN, Zhu QS (2012) Incremental collaborative filtering recommender based on regularized matrix factorization. Knowl-Based Syst 27:271–280

Manning CD, Raghavan P, Schütze H (2008) Introduction to information retrieval. Cambridge University Press

McCarthy K, Reilly J, McGinty L, Smyth B (2004) Thinking positively-explanatory feedback for conversational recommender systems. In: Proceedings of the European conference on case-based reasoning (ECCBR-04) explanation workshop, pp 115–124

Meng DY, Torre FDL (2013) Robust matrix factorization with unknown noise. In: ICCV, pp 1337–1344

Min F, Hu QH, Zhu W (2014) Feature selection with test cost constraint. Int J Approx Reason 55(1):167–179

Paradarami TK, Bastian ND, Wightman JL (2017) A hybrid recommender system using artificial neural networks. Expert Syst Appl 83:300–313

Qian FL, Min QQ, Zhao S, Chen J, Wang XY, Zhang YP (2019) Three-way decision collaborative recommendation algorithm based on user reputation. In: IJCRS, pp 424–438

Smyth B, McGinty L, Reilly J, McCarthy K (2004) Compound critiques for conversational recommender systems. In: Proceedings of IEEE/WIC/ACM international conference on web intelligence (WI'04), pp 145–151

Sun YM, Zhang Y (2018) Conversational recommender system. In: SIGIR, pp 235–244

Tapkan P, Özbakır L, Kulluk S, Baykasoğlu A (2016) A cost-sensitive classification algorithm: bee-miner. Knowl-Based Syst 95:99–113

Xu YY, Zhang HR, Min F (2017) A three-way recommender system for popularity-based costs. In: IJCRS, pp 278–289

Yang ZY, Zhang Y, Xiang Y, Yan W, Xie SL (2020) Non-negative matrix factorization with dual constraints for image clustering. IEEE Trans Syst, Man, Cybern: Syst 50(7):2524–2533

Yao YY (2010) Three-way decisions with probabilistic rough sets. Inf Sci 180(3):341–353

Yao LN, Sheng QZ, Ngu AH, Yu J, Segev A (2014) Unified collaborative and content-based web service recommendation. IEEE Trans Serv Comput 8(3):453–466

Ye XQ, Liu D (2021) An interpretable sequential three-way recommendation based on collaborative topic regression. Expert Syst Appl 168:114454

Yin HZ, Cui B, Li J, Yao JJ, Chen C (2012) Challenging the long tail recommendation. In: Proceedings of the VLDB endowment, pp 896–907

Zhang JY, Pu P (2006) A comparative study of compound critique generation in conversational recommender systems. In: Proceedings of international conference on adaptive hypermedia and adaptive web-based systems, pp 234–243

Zhang HR, Min F (2016) Three-way recommender systems based on random forests. Knowl-Based Syst 91:275–286

Zhang HR, Min F, Shi B (2017) Regression-based three-way recommendation. Inf Sci 378:444–461

Zhang QH, Yang CC, Wang GY (2021) A sequential three-way decision model with intuitionistic fuzzy numbers. IEEE Trans Syst, Man, Cybern: Syst 51(5):2640–2652

Zhao XX, Zhang WN, Wang J (2013) Interactive collaborative filtering. In: CIKM, pp 1411–1420

Zhou B, Yao YY, Luo JG (2010) A three-way decision approach to email spam filtering. In: Proceedings of the 23rd Canadian conference on artificial intelligence, pp 28–39