Mitigating Data Sparsity Using Similarity Reinforcement-Enhanced Collaborative Filtering

YAN HU, University of Chinese Academy of Sciences and Wayne State University
WEISONG SHI, Wayne State University
HONG LI, Institute of Information Engineering, Chinese Academy of Sciences
XIAOHUI HU, Institute of Software, Chinese Academy of Sciences

The data sparsity problem has attracted significant attention in collaborative filtering-based recommender systems. To alleviate data sparsity, several previous efforts employed hybrid approaches that incorporate auxiliary data sources into recommendation techniques, like content, context, or social relationships. However, due to privacy and security concerns, it is generally difficult to collect such auxiliary information. In this article, we focus on the pure collaborative filtering methods without relying on any auxiliary data source. We propose an improved memory-based collaborative filtering approach enhanced by a novel similarity reinforcement mechanism. It can discover potential similarity relationships between users or items by making better use of known but limited user-item interactions, thus to extract plentiful historical rating information from similar neighbors to make more reliable and accurate rating predictions. This approach integrates user similarity reinforcement and item similarity reinforcement into a comprehensive framework and lets them enhance each other. Comprehensive experiments conducted on several public datasets demonstrate that, in the face of data sparsity, our approach achieves a significant improvement in prediction accuracy when compared with the state-of-the-art memory-based and model-based collaborative filtering algorithms.

CCS Concepts: • **Information systems → Collaborative filtering**; **Nearest-neighbor search**; **Social recommendation;**

Additional Key Words and Phrases: Recommender system, rating prediction, personalization, data sparsity, similarity reinforcement

ACM Reference Format:
Yan Hu, Weisong Shi, Hong Li, and Xiaohui Hu. 2017. Mitigating data sparsity using similarity reinforcement-enhanced collaborative filtering. ACM Trans. Internet Technol. 17, 3, Article 31 (June 2017), 20 pages.
DOI: http://dx.doi.org/10.1145/3062179

1. INTRODUCTION

Modern customers are faced with a variety of choices. Content providers and online retailers provide a huge selection of information and products, with unprecedented

This research is partially supported by the National Natural Science Foundation of China, under Grants U1435220 and 61503365.
Authors' addresses: Y. Hu, Institute of Software, Chinese Academy of Sciences, No.4, South Fourth Street, Zhongguancun, Haidian District, Beijing, China, and the Department of Computer Science, Wayne State University, 5057 Woodward Ave., Suite 14102.2, Detroit, MI; email: huyanlh@126.com; W. Shi, the Department of Computer Science, Wayne State University, 5057 Woodward Ave., Suite 14102.2, Detroit, MI; email: weisong@wayne.edu; H. Li (corresponding author), Institute of Information Engineering, Chinese Academy of Sciences, No.89, Minzhuang Road, Haidian District, Beijing, China; email: lihong@iie.ac.cn; X. Hu, Institute of Software, Chinese Academy of Sciences, No.4, South Fourth Street, Zhongguancun, Haidian District, Beijing, China; email: hxh@iscas.ac.cn.

DOI: http://dx.doi.org/10.1145/3062179

opportunities to satisfy all kinds of tastes and needs. Recommender systems [36, 37] provide an excellent solution for solving the problem of information overloading. They aim to study the relationships between users and items, reveal users' preferences on items, and recommend the most suitable items to users. Commercial applications of recommender systems include e-commerce (e.g., Amazon, eBay) [40], fashion [13, 44], food and restaurants [34], social events [31], and art (e.g., movies, music, books) [5, 26, 45]. In these areas, recommender systems providing excellent personalized suggestions significantly increase the likelihood that a consumer purchases a product or selects an item compared with unpersonalized suggestions.

Collaborative Filtering (CF) [15] is one of the most commonly used technologies in recommender systems. The intuition behind CF is that similar users share similar interests and preferences (user-based) and similar items have similar characteristics (item-based). User-based methods find similar neighbors for an active user by studying the similarities between users' preferences. After that, recommender systems predict the unknown preferences of the active user on a set of candidate items and generate a ranking list of items, that the active user will like the most. Item-based methods reveal relationships between items and use historical information from similar items to make preference predictions. The biggest advantage of CF is that it does not rely on the content and form of items and can be applied to items without textual description.

Generally speaking, there are two types of CF algorithms: memory-based [8, 38, 48, 49, 57] and model-based [2, 6, 17, 43]. Memory-based algorithms construct a user-item rating matrix based on collected rating data. Each element of the matrix represents a user's rating on an item. The algorithms attempt to discover relationships between users or items by calculating similarities between users or items. After that, they predict the unknown preference of an active user based on the past information provided by similar users and items. Model-based algorithms either explore a "latent space" or build a model to capture the user-item relationships. They use existing data to train the pre-defined model and make preference predictions based on the trained model. Regression [46], matrix factorization [28], and Bayesian methods [16] are popular approaches that fall into this category. They have a good theoretical basis, but are not so intuitive in terms of interpreting prediction results as memory-based algorithms are [49]. This article focuses on memory-based CF algorithms.

It has been demonstrated that memory-based CF algorithms have achieved considerable success in recommender systems. However, they suffer from the data sparsity problem [10]. In real-world applications, most users only rate a small number of items and most items are only rated by a small number of users, so the rating matrix is quite sparse. In this case, it has been proved that traditional similarity measures, such as Pearson Correlation Coefficient (PCC) and Cosine-based similarity, cannot function well [29]. Therefore, the data sparsity problem leads to great difficulties in finding similar neighbors, and prevents memory-based CF algorithms from making accurate preference predictions. This problem was partially solved by incorporating auxiliary data sources, which are relatively dense, such as the content of items [50], users' locations [53], and users' friendship or social trust information [14, 18–20, 30]. However, the introduction of auxiliary information complicates system implementation. Additionally, due to privacy and security concerns, users may not be willing to provide their personal information, such as their locations or social relationships. In this article, we focus on a pure memory-based CF approach and propose a novel similarity reinforcement mechanism to mitigate the data sparsity problem. Our approach does not require any auxiliary information. It only relies on known user-item interactions and yet achieves significant improvement on prediction accuracy compared with the state-of-the-art CF algorithms. The key contributions of this work are summarized as follows:

—A novel similarity reinforcement mechanism is proposed to enhance the memory-based CF algorithm so that more potential similarity relationships can be discovered between users or items, and more historical information can be extracted from similar neighbors to lessen data sparsity.
—A comprehensive algorithm is designed to integrate user similarity reinforcement and item similarity reinforcement and let the two procedures enhance each other.
—The effectiveness of the proposed approach to handle data sparsity has been validated by the experiments conducted on three publicly available datasets: MovieLens 1M[1], Netflix[2], and Yahoo! Music[3].

The rest of the article is organized as follows. Section 2 introduces some related work. In Section 3, we present the basic notations used throughout the article and the formal problem definition. In Section 4, a memory-based CF approach enhanced by the novel similarity reinforcement mechanism is proposed to alleviate the data sparsity problem in recommender systems. Section 5 conducts comprehensive experiments to validate the effectiveness of our approach. Section 6 discusses the computational complexity of the proposed approach. Finally, conclusions are drawn in Section 7.

2. RELATED WORK

CF technology has been developed since the early 1990s. It was employed in the Tapestry system [9] to filter mails from several mailing lists by analyzing the reactions (annotations) given by collaborative users to documents they read. Since then, many CF algorithms have been proposed. They are generally categorized into two classes: memory-based and model-based. Memory-based approaches utilize historical user-item rating data to estimate the similarities between users or items. Based on that, historical rating information from similar neighbors are collected to predict the preferences of an active user on a set of candidate items. The most popular memory-based CF approaches include user-based [3, 21], item-based [7, 38], and hybrid [56, 57] algorithms. User-based algorithms use the historical rating data from similar users for preference prediction, while item-based methods extract data from similar items for prediction. The hybrid CF approach integrates the two former methods, and achieves a better prediction performance. Model-based CF algorithms construct appropriate models (e.g., matrix factorization models [25, 41], clustering models [43], and latent factor models [4]) for user-item relationships. Model training is generally performed offline to determine the optimal values of model parameters. Finally, the trained models are used for preference prediction.

In recent years, based on traditional CF approaches, some new methods like feature engineering and multi-layered neural networks were proposed for recommender systems. Rendle et al. [35] argued that applying factorization approaches to a new prediction problem is a nontrivial task and requires a lot of expert knowledge. Therefore, they proposed factorization machines (FM), a generic and easy-to-use approach. Factorization machines can mimic most factorization models by feature engineering and combine the generality and flexibility of feature engineering with the high prediction accuracy of factorization models. Wu et al. [51] used a three-layer neural network to reconstruct the full user-item preferences by learning the latent representations of the partially observed user-item rating data. They also proved that the proposed model is a generalization of several well-known collaborative filtering models, like latent

[1]http://grouplens.org.
[2]http://www.netflixprize.com.
[3]http://webscope.sandbox.yahoo.com/#datasets.

factor model, similarity model, and factorized similarity model, but with more flexible components.

Although CF has achieved significant success in recommender systems, it still has the problems of scalability and data sparsity when applied in the real world. Wu et al. [52] tried to deal with the problem of scalability by clustering users and items into several subgroups. Each subgroup includes a set of like-minded users and their liked items. Traditional CF methods are applied to each subgroup, and the recommendation results from all subgroups can be easily aggregated. Zhang et al. [55] presented a Localized Matrix Factorization (LMF) framework. They attempted to overcome the challenges of scalability and sparsity by transforming a large sparse matrix into Recursive Bordered Block Diagonal Form (RBBDF) and extracting smaller and dense submatrices from RBBDF. This approach has the potential to improve prediction accuracy by factorizing these submatrices independently. This framework is also suitable for parallelization and thus improves system scalability. Jing et al. [22] proposed a sparse probabilistic matrix factorization (SPMF) method by using the Laplacian distribution to model the user/item factor vector. The Laplacian distribution has the ability to generate sparse coding. Additionally, the heavy tails in the Laplacian distribution help to identify the tail items and partially solve the long tail problem.

Other studies have used auxiliary information to deal with the data sparsity problem, including item semantics and content information, or users' social relationships (e.g., friendship or trust). Moshfeghi et al. [32] imported emotion and semantic features related to items into recommender systems. They extracted two emotion spaces from movie plot summaries and review data, and considered three semantic spaces: actor, director, and genre. Additionally, they employed a framework which relies on an extended Latent Dirichlet Allocation (LDA) model for prediction. They concluded that actor and movie spaces are the least and most sensitive spaces to dataset size and data sparsity, and the model that incorporates all spaces performs the best at remedying data sparsity compared with state-of-the-art CF systems. Wang et al. [47] proposed a hierarchical Bayesian model called Collaborative Deep Learning (CDL), which jointly performs deep representation learning for item content and collaborative filtering for the rating matrix. By performing deep learning collaboratively, CDL can simultaneously extract an effective deep feature representation from content and capture similarities and implicit relationships between items or users. Pan et al. [33] attempted to handle the issue of data sparsity in a target domain by transferring knowledge about both users and items from other related auxiliary data sources that are relatively dense. They used a matrix-based transfer learning technology, which is capable of dealing with the data heterogeneity in different domains.

Kaya and Alpaslan [23] extracted like-minded ideas from social networks to mitigate the problem of data sparsity. Furthermore, their study verified that social networks correlated with a specific domain carry better information and knowledge required to make recommendations in that specific domain. Konstas et al. [24] considered for music track recommendation both the social annotations (tags) and friendship relations in a social graph constructed among users, items, and tags. Because of the vagueness of friendship, friends may show different opinions on items. However, users participating in the same actual or virtual communities are more inclined to share similar interests [54]. Therefore, the data extracted from both friendship and membership are leveraged and fused for more accurate preference prediction.

Compared with membership and friendship, trust information is less vague and can better facilitate the discovery of similarity relationships. In research literature, trust information has been adopted in many recommendation approaches. Haydar et al. [14] combined opinion similarity and trust similarity to increase the connectivity between users, and reduced the impact of data sparsity. Guo et al. [11, 12] utilized the merged

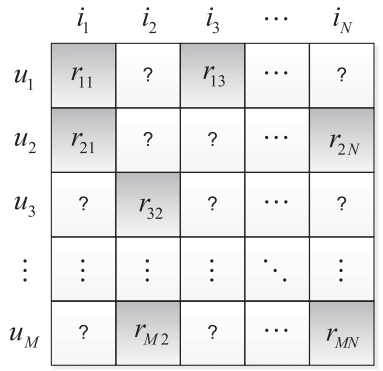

Fig. 1. Rating matrix.

ratings from trusted neighbors to compensate for the incomplete preference of an active user. Jamali and Ester [18] designed a random walk mechanism to combine the trust-based and item-based CF approaches for recommendation. Additionally, they applied the trust propagation technique to the matrix factorization models in a subsequent study [19]. Ma et al. [30] designed a novel probabilistic factor analysis model, in which user favors and the preferences of their trusted friends are well merged together.

One drawback of the approaches based on auxiliary sources is the high dependency on auxiliary information. It increases the storage cost and complicates the implementation of recommender systems. Furthermore, due to privacy and security concerns, it is generally difficult to collect users' trust, friendship, or membership information. Hence, it is important to design effective algorithms only based on the rating information for higher-quality recommendation. Liu et al. [27] focused on memory-based CF and proposed an improved heuristic similarity measure only based on users' rating data. The model provides a more accurate similarity measure by taking into account several factors of similarity, namely Proximity, Impact, Popularity, and users' rating preferences. However, it still relies on the co-rated items between users. If two users have no items rated in common, their similarities still cannot be estimated. In this article, we focus on a pure memory-based CF algorithm without using any auxiliary data source. We designed a novel similarity reinforcement mechanism, that can reveal implicit similarities between users or items. Compared with traditional CF methods, our method improves the prediction accuracy significantly and alleviates the data sparsity problem effectively.

3. NOTATION AND PROBLEM DEFINITION

Before presenting the proposed approach, we first introduce the basic notations and concepts used throughout the article and then formally describe the problem to be solved in recommender systems.

We use $[N]$ to denote a set of consecutive positive integers $\{1, 2, \ldots, N\}$. Scalars are denoted by lowercase letters such as a. Vectors are denoted by boldface lowercase letters such as $\boldsymbol{u}$. Moreover, we use boldface uppercase letters such as $\boldsymbol{M}$ to denote matrices. The Frobenius norm of a matrix $\boldsymbol{M} \in \mathbb{R}^{m \times n}$ is denoted by $\|\boldsymbol{M}\|_{\mathrm{F}}$, i.e, $\|\boldsymbol{M}\|_{\mathrm{F}} = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |\boldsymbol{M}_{ij}|^2}$.

In a recommender system, there is a set of M users $\mathcal{U} = \{u_1, u_2, \ldots, u_M\}$ and a set of N items $\mathcal{I} = \{i_1, i_2, \ldots, i_N\}$. The interactions between users and items are summarized in a $M \times N$ rating matrix $\boldsymbol{R}$, as depicted in Figure 1. Each row of the matrix corresponds

to a user and each column corresponds to an item. The (p, q) entry r_{pq} ($p \in [M], q \in [N]$) in the matrix denotes the rating value given by user u_p to item i_q. If a user did not rate an item, the corresponding entry is called a missing value.

Traditional memory-based CF algorithms utilize the observed values in the rating matrix to predict missing values. When a missing rating value is predicted for user u_p on item i_q, u_p and i_q are called an active user and a candidate item, respectively. A typical memory-based CF recommendation problem involves three main steps. In the first step, the similarities between users or items are estimated using some similarity measures, such as PCC and Cosine-based similarities. In the second step, the algorithm finds a set of similar users for the active user and a set of similar items for the candidate item, where similar users/items are those with positive similarities. In the third step, the preferences of similar users on the candidate item and the preferences of the active user on similar items are combined to predict the missing rating value for the active user on the candidate item. In the rest of the article, we interchangeably use the terms "similar users/items" and "users/items with positive similarities."

However, we note that the rating matrix is only partially filled with observed values. The percentage of missing values in a matrix is called the *sparsity level* of the matrix. The sparsity level of the user-item rating matrix in real-world recommender systems usually exceeds 90%. This brings great challenges to similarity calculation and rating prediction within memory-based CF algorithms. In the following section, we propose a novel similarity reinforcement approach to reduce the impact of data sparsity on recommender systems.

4. SIMILARITY REINFORCEMENT-ENHANCED COLLABORATIVE FILTERING

In this section, we first explain the underlying motivations and assumptions of our approach and then present the proposed approach, named Similarity Reinforcement-Enhanced Collaborative Filtering (SRCF).

4.1. Similarity Computation

As mentioned previously, the similarity computing between users or items is a core issue in memory-based CF algorithms. Well-known similarity measures include PCC, constrained PCC, Cosine-based similarity, adjusted Cosine-based similarity, and other improved versions [1]. To introduce our approach, we use the PCC similarity as an example and our techniques can be applied to other similarity measures in a similar way. In user-based CF methods, the PCC similarity between two users u_a and u_b is calculated by the following equation:

$$sim'(u_a, u_b) = \frac{\sum_{i_k \in \mathcal{I}_{ab}} (r_{ak} - \bar{r}_a)(r_{bk} - \bar{r}_b)}{\sqrt{\sum_{i_k \in \mathcal{I}_{ab}} (r_{ak} - \bar{r}_a)^2} \sqrt{\sum_{i_k \in \mathcal{I}_{ab}} (r_{bk} - \bar{r}_b)^2}}, \tag{1}$$

where $\mathcal{I}_{ab} = \mathcal{I}_a \cap \mathcal{I}_b$ is the subset of items rated by both u_a and u_b, and $\bar{r}_a$ and $\bar{r}_b$ are the average ratings given by u_a and u_b, respectively. With this definition, the similarity between two users is in the interval $[-1,1]$, with a larger PCC value indicating a higher similarity value. Likewise, in item-based CF methods, the PCC similarity between two items i_k and i_l is computed by:

$$sim'(i_k, i_l) = \frac{\sum_{u_a \in \mathcal{U}_{kl}} (r_{ak} - \bar{r}_k)(r_{al} - \bar{r}_l)}{\sqrt{\sum_{u_a \in \mathcal{U}_{kl}} (r_{ak} - \bar{r}_k)^2} \sqrt{\sum_{u_a \in \mathcal{U}_{kl}} (r_{al} - \bar{r}_l)^2}}, \tag{2}$$

where $\mathcal{U}_{kl} = \mathcal{U}_k \cap \mathcal{U}_l$ is the subset of users who rated both i_k and i_l, and $\bar{r}_k$ and $\bar{r}_l$ are the average ratings of items i_k and i_l, respectively.

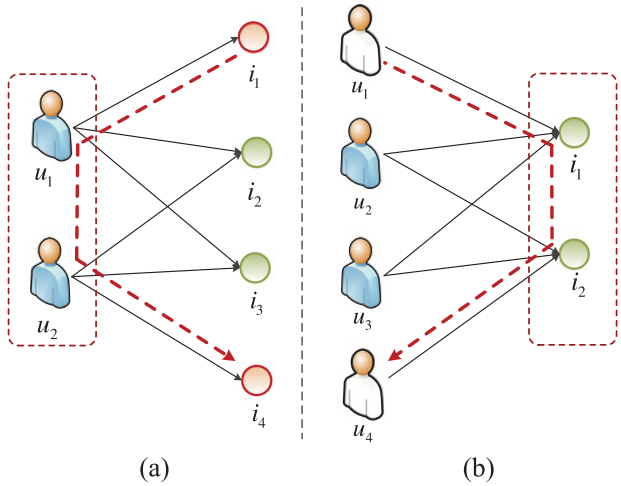

Fig. 2. An example for similarity reinforcement.

Equation (1) often overestimates the similarities between two users who are not similar actually but happen to give similar ratings to a small number of co-rated items. To deal with this problem, a similarity weight [56, 57] is utilized to reduce the impact of a small number of co-rated items:

$$sim(u_a, u_b) = \frac{2|\mathcal{I}_{ab}|}{|\mathcal{I}_a| + |\mathcal{I}_b|} \cdot sim'(u_a, u_b), \tag{3}$$

where $|\mathcal{I}_{ab}| = |\mathcal{I}_a \cap \mathcal{I}_b|$ is the number of items that u_a and u_b both rated, $|\mathcal{I}_a|$ and $|\mathcal{I}_b|$ are the numbers of items rated by u_a and u_b, respectively. Similarly, item similarity is also revised to reduce the impact of a small number of common users:

$$sim(i_k, i_l) = \frac{2|\mathcal{U}_{kl}|}{|\mathcal{U}_k| + |\mathcal{U}_l|} \cdot sim'(i_k, i_l), \tag{4}$$

where $|\mathcal{U}_{kl}| = |\mathcal{U}_k \cap \mathcal{U}_l|$ is the number of users who rated both i_k and i_l, $|\mathcal{U}_k|$ and $|\mathcal{U}_l|$ are the numbers of users who rated i_k and i_l, respectively.

4.2. Similarity Reinforcement

4.2.1. Motivations. As indicated by Equations (1) to (4), if two users/items have no item/user intersection, namely $\mathcal{I}_{ab} = null$ or $\mathcal{U}_{kl} = null$, their similarity cannot be calculated. Unfortunately, due to the huge number of users and items in a recommender system, even very active users may rate only a few items and even very popular items may be rated by only a few users, so the rating matrix $\boldsymbol{R}$ is quite sparse. Therefore, the traditional similarity measures cannot identify enough similar neighbors for active users or candidate items, which dramatically degrades the performance of CF algorithms. To address the problem, we designed a novel similarity reinforcement mechanism to compensate for the deficiency of traditional memory-based CF approaches. Our method aims to reveal implicit similarity relationships between users or items without any auxiliary information. Implicit similarity relationships can help to get more historical data from similar neighbors and achieve higher prediction accuracy.

The similarity reinforcement mechanism is mainly based on PCC similarities and observed user-item ratings. A toy example is shown in Figure 2. In Figure 2(a), there are two users u_1 and u_2 and four items i_1 to i_4. u_1 and u_2 both rated i_2 and i_3, while i_1 was only rated by u_1 and i_4 was only rated by u_2. According to traditional similarity

	i_1	i_2	i_3	i_4	i_5
u_1	5	5	5	--	--
u_2	5	5	--	5	1

(a)

	i_1	i_2	i_3	i_4	i_5
u_1	5	1	5	--	--
u_2	1	5	--	5	1

(b)

Fig. 3. Similarity propagation.

measures, the similarity between i_1 and i_4 cannot be calculated, since the two items are not rated by any common user. However, we observe that i_1 is rated by u_1, and i_4 is rated by u_2, and u_1 and u_2 have a PCC similarity since they have two items i_2 and i_3 rated in common. Therefore, i_1 and i_4 can establish an implicit similarity relationship through the affinity between u_1 and u_2. Likewise, as shown in Figure 2(b), u_1 and u_4, who have no item rated in common, can build an implicit similarity relationship through the affinity between i_1 and i_2. Therefore, our similarity reinforcement mechanism is premised on the following three assumptions:

—Users with no direct PCC similarities can build implicit similarities through the affinity between their items.
—Items with no direct PCC similarities can build implicit similarities through the affinity between their users.
—User similarity reinforcement and item similarity reinforcement can enhance each other.

The next problem is how similarities propagate between users or items without direct PCC similarities. We use two examples in Figure 3 to illustrate our basic idea. We assume that users' ratings range from 1 to 5, where 1 means "dislike very much" and 5 means "like very much." In Figure 3(a), u_1 and u_2 give a similar rating of 5 to their co-rated items i_1 and i_2. As a result, u_1 and u_2 have a positive similarity, as do i_1 and i_2. On the other hand, u_1 rates i_3 as 5 and u_2 rates i_4 as 5. Although i_3 and i_4 have no common user, it is reasonable to establish an implicit positive similarity between i_3 and i_4, since they receive similar ratings from two similar users, respectively. As for i_3 and i_5, they receive totally different ratings from u_1 and u_2, respectively, so a negative similarity should be built between i_3 and i_5. In Figure 3(b), u_1 rates i_1 and i_2 as 1 and 5, while u_2 rates i_1 and i_2 as 5 and 1. As a result, u_1 and u_2 have a negative PCC similarity, as do i_1 and i_2. As for the item pair i_3 and i_4, they have no common user, and u_1 rates i_3 as 5 and u_2 rates i_4 as 5. We can infer that i_3 is similar to i_1 because the same user u_1 rates both of them as 5, and i_4 is similar to i_2 because the same user u_2 rates both of them as 5. However, i_1 is dissimilar to i_2, so we can infer that i_3 and i_4 are also dissimilar. As for i_3 and i_5, i_5 receives a rating of 1 from u_2, so i_5 is similar to i_1 since they both receive a rating of 1 from the same user u_2, and we can further infer that i_5 is similar to i_3 because they are both similar to i_1.

Based on the above analysis, we summarize four ways of similarity propagation between items without user intersection: (i) if two users with a positive similarity give similar ratings to two items, the items have a positive similarity; (ii) if two users with a positive similarity give dissimilar ratings to two items, the items have a negative similarity; (iii) if two users with a negative similarity give similar ratings to two items, the items have a negative similarity; (iv) if two users with a negative similarity give dissimilar ratings to two items, the items have a positive similarity. The similarity propagation mechanism between users without item intersection is analogous.

4.2.2. Similarity Reinforcement Algorithm. Suppose u_a and u_b are two users. The item set rated by u_a is denoted by $\mathcal{I}_a = \{i_1^a, i_2^a, \ldots, i_P^a\}$ ($\mathcal{I}_a \subset \mathcal{I}$), and the item set rated by u_b is denoted by $\mathcal{I}_b = \{i_1^b, i_2^b, \ldots, i_Q^b\}$ ($\mathcal{I}_b \subset \mathcal{I}$). We define two sets $\mathcal{PI}_{ab} = \{(i_p^a, i_q^b) \mid (i_p^a, i_q^b) \in \mathcal{I}_a \times \mathcal{I}_b, sim(i_p^a, i_q^b) > 0\}$, which includes the item pairs from $\mathcal{I}_a \times \mathcal{I}_b$ with positive similarities, and $\mathcal{NI}_{ab} = \{(i_p^a, i_q^b) \mid (i_p^a, i_q^b) \in \mathcal{I}_a \times \mathcal{I}_b, sim(i_p^a, i_q^b) < 0\}$, which consists of the item pairs from $\mathcal{I}_a \times \mathcal{I}_b$ with negative similarities. Accordingly, the similarity reinforcement between u_a and u_b can be achieved based on the affinity between each pair of items (i_p^a, i_q^b) from $\mathcal{PI}_{ab}$ or $\mathcal{NI}_{ab}$:

$$sim(u_a, u_b) = (1 - \alpha) \cdot sim(u_a, u_b) + \alpha \cdot \frac{\sum_{(i_p^a, i_q^b) \in \mathcal{PI}_{ab} \cup \mathcal{NI}_{ab}} w_{pq} \cdot sim(i_p^a, i_q^b)}{\sum_{(i_p^a, i_q^b) \in \mathcal{PI}_{ab} \cup \mathcal{NI}_{ab}} |w_{pq}|}, \tag{5}$$

where α ($0 \leq \alpha \leq 1$) is a damping factor controlling the degree that the reinforced user similarity relies on its previous value or the contributions of item similarities. At the beginning, the user pair (u_a, u_b) may or may not have direct PCC similarity. If not, $sim(u_a, u_b)$ should only depend on the contributions of item similarities at the first step, i.e., $\alpha = 1$. After that, α should be set to its original value, a damping factor which is bigger than 0 and smaller than 1. Additionally, the parameter w_{pq} ($p \in [P]$, $q \in [Q]$) weighs the contribution of the item pair (i_p^a, i_q^b). w_{pq} is regarded as the similarity between two ratings and defined as a linear function of the absolute rating difference $d(r_{ai_p^a}, r_{bi_q^b})$:

$$w_{pq} = 1 - 2d(r_{ai_p^a}, r_{bi_q^b}) = 1 - 2|r_{ai_p^a} - r_{bi_q^b}|, \tag{6}$$

where $r_{ai_p^a}$ is the rating value given by user u_a to item i_p^a, and $r_{bi_q^b}$ is the rating value given by user u_b to item i_q^b. All ratings should be first normalized by the following equation:

$$r_{nor} = \frac{r\_r_{min}}{r_{max} - r_{min}}, \tag{7}$$

where r_{max} and r_{min} are the maximum and minimum rating values before normalization. After normalization, the absolute rating difference $|r_{ai_p^a} - r_{bi_q^b}|$ falls within the interval [0,1]. We split this interval [0,1] into two subintervals, [0,0.5] and (0.5,1]. If $|r_{ai_p^a} - r_{bi_q^b}|$ ranges from 0 to 0.5, the two ratings $r_{ai_p^a}$ and $r_{bi_q^b}$ are considered similar ($0 \leq w_{pq} \leq 1$), with a smaller value of $|r_{ai_p^a} - r_{bi_q^b}|$ indicating a larger value of w_{pq}. Given $w_{pq} \geq 0$, if the item pair (i_p^a, i_q^b) comes from $\mathcal{PI}_{ab}$, we derive $w_{pq} \cdot sim(i_p^a, i_q^b) \geq 0$. It means that similar ratings given to similar items contribute positively to the user similarity reinforcement. Otherwise, if (i_p^a, i_q^b) comes from $\mathcal{NI}_{ab}$, we derive $w_{pq} \cdot sim(i_p^a, i_q^b) \leq 0$, which means that similar ratings given to dissimilar items contribute negatively to the user similarity reinforcement. On the other hand, if $|r_{ai_p^a} - r_{bi_q^b}|$ ranges from 0.5 to 1, the two ratings are regarded as dissimilar ($-1 \leq w_{pq} < 0$). In this case, if the item pair (i_p^a, i_q^b) comes from $\mathcal{PI}_{ab}$, we derive $w_{pq} \cdot sim(i_p^a, i_q^b) < 0$. It means that dissimilar ratings given to similar items also contribute negatively to the user similarity reinforcement. Otherwise, if (i_p^a, i_q^b) comes from $\mathcal{NI}_{ab}$, we derive $w_{pq} \cdot sim(i_p^a, i_q^b) > 0$, which indicates that dissimilar ratings given to dissimilar items also contribute positively to the user similarity reinforcement.

Likewise, for two items i_k and i_l, the users who rated i_k and i_l are denoted by the two sets $\mathcal{U}_k = \{u_1^k, u_2^k, \ldots, u_C^k\}$ ($\mathcal{U}_k \subset \mathcal{U}$) and $\mathcal{U}_l = \{u_1^l, u_2^l, \ldots, u_D^l\}$ ($\mathcal{U}_l \subset \mathcal{U}$), respectively. Taking into account the contributions of users with positive or negative similarities, we define two sets, $\mathcal{PU}_{kl} = \{(u_c^k, u_d^l) \mid (u_c^k, u_d^l) \in \mathcal{U}_k \times \mathcal{U}_l, sim(u_c^k, u_d^l) > 0\}$, which includes all

user pairs from $\mathcal{U}_k \times \mathcal{U}_l$ with positive similarities, and $\mathcal{NU}_{kl} = \{(u_c^k, u_d^l) \mid (u_c^k, u_d^l) \in \mathcal{U}_k \times \mathcal{U}_l, sim(u_c^k, u_d^l) < 0\}$, which consists of user pairs from $\mathcal{U}_k \times \mathcal{U}_l$ with negative similarities. Consequently, the similarity between i_k and i_l can be reinforced as:

$$sim(i_k, i_l) = (1-\alpha) \cdot sim(i_k, i_l) + \alpha \cdot \frac{\sum_{(u_c^k, u_d^l) \in \mathcal{PU}_{kl} \cup \mathcal{NU}_{kl}} w_{cd} \cdot sim(u_c^k, u_d^l)}{\sum_{(u_c^k, u_d^l) \in \mathcal{PU}_{kl} \cup \mathcal{NU}_{kl}} |w_{cd}|}, \tag{8}$$

where w_{cd} ($c \in [C],\ d \in [D]$) is the weight related to the user pair (u_c^k, u_d^k), which is used to weigh the contribution of the similarity between u_c^k and u_d^l. Formally, w_{cd} is defined by:

$$w_{cd} = 1 - 2\mathrm{d}(r_{u_c^k k}, r_{u_d^l l}) = 1 - 2|r_{u_c^k k} - r_{u_d^l l}|, \tag{9}$$

where $r_{u_c^k k}$ is the normalized rating value given by user u_c^k to item i_k, and $r_{u_d^l l}$ is that given by user u_d^l to item i_l.

4.2.3. Mutually Enhancing Similarity Reinforcement Algorithm for Both Users and Items. The similarity reinforcements for users and items are not isolated from each other. They can enhance each other. As shown in Equations (5) and (8), the user similarity reinforcement depends on item similarities, and vice versa. Therefore, we designed a comprehensive algorithm to incorporate the two processes into one and let them enhance each other. The entire procedure of the *Comprehensive Similarity Reinforcement* algorithm, or "*CSR*" for short, is summarized in Algorithm 1. In this algorithm, $\boldsymbol{U}_{re\_sim}$ and $\boldsymbol{I}_{re\_sim}$ denote the reinforced user and item similarity matrices. $\boldsymbol{U}_{re\_sim\_last}$ and $\boldsymbol{I}_{re\_sim\_last}$ record the values of $\boldsymbol{U}_{re\_sim}$ and $\boldsymbol{I}_{re\_sim}$ from the last iteration step. Line 1 calculates the traditional PCC-based user and item similarities. Lines 2 and 3 perform initialization. The iteration procedure is shown in lines 4 through 24. Within the iteration, lines 5 and 6 record the reinforced similarity matrices from the last step, and the user similarity reinforcement is performed by lines 7 through 12, and the item similarity reinforcement by lines 13 through 18. The iteration ends when the maximal number of iterations is reached (line 4) or the variations of the user similarity matrix and the item similarity matrix drop below a predefined threshold ϵ (lines 19 through 23). Since similarity matrices are symmetric, we only need to perform one computation for each pair of symmetric elements in a similarity matrix, as shown in lines 10 and 16. The last line returns the reinforced user and item similarity matrices. In this algorithm, we store the similarity matrices on secondary storage. While performing similarity reinforcement for every pair of users and every pair of items, only the data involved in the present computation are loaded into memory. The computational complexity of the algorithm will be discussed later in Section 6.

4.3. Rating Prediction

In this section, we describe how we use the historical ratings from similar users and items, namely users and items with positive similarities, to make rating predictions. Suppose that u_a is an active user and i_k is a candidate item. u_b and i_l are the similar neighbors of u_a and i_k, respectively. The user-based and item-based rating prediction results for the active user u_a on the candidate item i_k are presented by the following two equations:

$$\hat{r}_{ak}^u = \bar{r}_a + \frac{\sum_{u_b \in \mathcal{N}_{u_a}} \boldsymbol{U}_{re\_sim}(a, b) \cdot (r_{bk} - \bar{r}_b)}{\sum_{u_b \in \mathcal{N}_{u_a}} \boldsymbol{U}_{re\_sim}(a, b)}, \tag{10}$$

ALGORITHM 1: *CSR* Algorithm

Input: user-item rating matrix $\boldsymbol{R}$, damping factor α, the maximum number of iteration $iter\_max$, threshold ϵ.
Output: reinforced user similarity matrix $\boldsymbol{U}_{re\_sim}$, reinforced item similarity matrix $\boldsymbol{I}_{re\_sim}$.

compute user similarity matrix $\boldsymbol{U}_{sim}$ and item similarity matrix $\boldsymbol{I}_{sim}$ based on PCC;
$\boldsymbol{U}_{re\_sim} \leftarrow \boldsymbol{U}_{sim}$;
$\boldsymbol{I}_{re\_sim} \leftarrow \boldsymbol{I}_{sim}$;
for $iter\_cur \leftarrow 1$ *to* $iter\_max$ **do**
  $\boldsymbol{U}_{re\_simlast} \leftarrow \boldsymbol{U}_{re\_sim}$;
  $\boldsymbol{I}_{re\_simlast} \leftarrow \boldsymbol{I}_{re\_sim}$;
  for $a \leftarrow 1$ *to* M **do**
    for $b \leftarrow a+1$ *to* M **do**
      Update $\boldsymbol{U}_{re\_sim}(a,b)$ according to Equation (5);
      $\boldsymbol{U}_{re\_sim}(b,a) \leftarrow \boldsymbol{U}_{re\_sim}(a,b)$;
    end
  end
  for $k \leftarrow 1$ *to* N **do**
    for $l \leftarrow k+1$ *to* N **do**
      Update $\boldsymbol{I}_{re\_sim}(k,l)$ according to Equation (8);
      $\boldsymbol{I}_{re\_sim}(l,k) \leftarrow \boldsymbol{I}_{re\_sim}(k,l)$;
    end
  end
  $d_U = \|\boldsymbol{U}_{re\_sim} - \boldsymbol{U}_{re\_simlast}\|_{\mathrm{F}}$;
  $d_I = \|\boldsymbol{I}_{re\_sim} - \boldsymbol{I}_{re\_simlast}\|_{\mathrm{F}}$;
  if $d_U < \epsilon$ *and* $d_I < \epsilon$ **then**
    break;
  end
end
return $\boldsymbol{U}_{re\_sim}$ and $\boldsymbol{I}_{re\_sim}$;

and

$$\hat{r}^{i}_{ak} = \bar{r}_k + \frac{\sum_{i_l \in \mathcal{N}_{i_k}} \boldsymbol{I}_{re\_sim}(k,l) \cdot (r_{al} - \bar{r}_l)}{\sum_{i_l \in \mathcal{N}_{i_k}} \boldsymbol{I}_{re\_sim}(k,l)}, \tag{11}$$

where $\mathcal{N}_{u_a} = \{u_b \mid \boldsymbol{U}_{re\_sim}(a,b) > 0, a \neq b\}$ and $\mathcal{N}_{i_k} = \{i_l \mid \boldsymbol{I}_{re\_sim}(k,l) > 0, k \neq l\}$ are the sets of similar neighbors of u_a and i_k, respectively.

In order to merge the user-based and item-based predictions into a synthesized result, we employ two confidence weights con_u and con_i [56, 57] to balance the two prediction results. The confidence weights are defined by:

$$con_u = \sum\nolimits_{u_b \in \mathcal{N}_{u_a}} \frac{\boldsymbol{U}_{re\_sim}(a,b)}{\sum_{u_b \in \mathcal{N}_{u_a}} \boldsymbol{U}_{re\_sim}(a,b)} \cdot \boldsymbol{U}_{re\_sim}(a,b), \tag{12}$$

and

$$con_i = \sum\nolimits_{i_l \in \mathcal{N}_{i_k}} \frac{\boldsymbol{I}_{re\_sim}(k,l)}{\sum_{i_l \in \mathcal{N}_{i_k}} \boldsymbol{I}_{re\_sim}(k,l)} \cdot \boldsymbol{I}_{re\_sim}(k,l). \tag{13}$$

Finally, the missing rating value is predicted by:

$$\hat{r}_{ak} = h_u \cdot \hat{r}^{u}_{ak} + h_i \cdot \hat{r}^{i}_{ak}, \tag{14}$$

where h_u and h_i [56, 57] are the weights of user-based and item-based prediction results and defined by:

$$h_u = \frac{\lambda \cdot con_u}{\lambda \cdot con_u + (1-\lambda) \cdot con_i}, \tag{15}$$

and

$$h_i = \frac{(1-\lambda) \cdot con_i}{\lambda \cdot con_u + (1-\lambda) \cdot con_i}, \tag{16}$$

where λ ($0 \leq \lambda \leq 1$) is a parameter employed to determine how much the final prediction result depends on the user-based and item-based prediction results.

5. EVALUATION

In this section, we conduct comprehensive experiments on three publicly available datasets to evaluate the effectiveness of the proposed approach to alleviate data sparsity.

5.1. Dataset

The three publicly available datasets used in our experiments are MovieLens 1M, Netflix and Yahoo! Music. All the three datasets contain ratings in a 1 to 5 scale. MovieLens 1M contains 1,000,209 ratings by 6,040 users of total 3,900 movies. Netflix contains more than 100 million ratings by 480,189 customers of 17,770 movies. Yahoo! Music contains 311,704 ratings by 15,400 users of 1,000 songs. The sparsity levels of the three datasets are 95.8%, 98.8%, and 98%, respectively.

5.2. Evaluation Metric

We adopt two widely used metrics, Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE), to measure the prediction accuracy of CF algorithms. The two metrics are particularly useful in the context of evaluating prediction accuracy in off-line tests. The first measure (MAE) treats every error equally, while the second one (RMSE) emphasizes larger errors. Let $\mathcal{T}$ denote the set of ratings to be predicted, i.e., $\mathcal{T} = \{(p,q) \mid p \in [M], q \in [N], r_{pq} \text{ to be predicted}\}$. Then, the MAE and RMSE metrics are defined by:

$$MAE = \frac{1}{|\mathcal{T}|} \cdot \sum_{(p,q)\in\mathcal{T}} |\hat{r}_{pq} - r_{pq}|, \tag{17}$$

and

$$RMSE = \sqrt{\frac{1}{|\mathcal{T}|} \cdot \sum_{(p,q)\in\mathcal{T}} (\hat{r}_{pq} - r_{pq})^2}, \tag{18}$$

where r_{pq} and $\hat{r}_{pq}$ are the actual and predicted rating values, respectively. A smaller value of MAE or RMSE indicates higher prediction accuracy.

5.3. Performance Comparison

In the experiments, we studied the prediction accuracy of our approach under different data sparsity levels and compared our approach with five well-known CF methods. They are the User-based CF using PCC (UPCC) [3], the Item-based CF using PCC (IPCC) [38], the hybrid CF using PCC [57], the Regularized Matrix Factorization (RMF) [25] and the Regularized Matrix Factorization with Biases (BRMF) [41]. These methods have been widely used as baseline methods in the study of recommender systems [39, 42, 49].

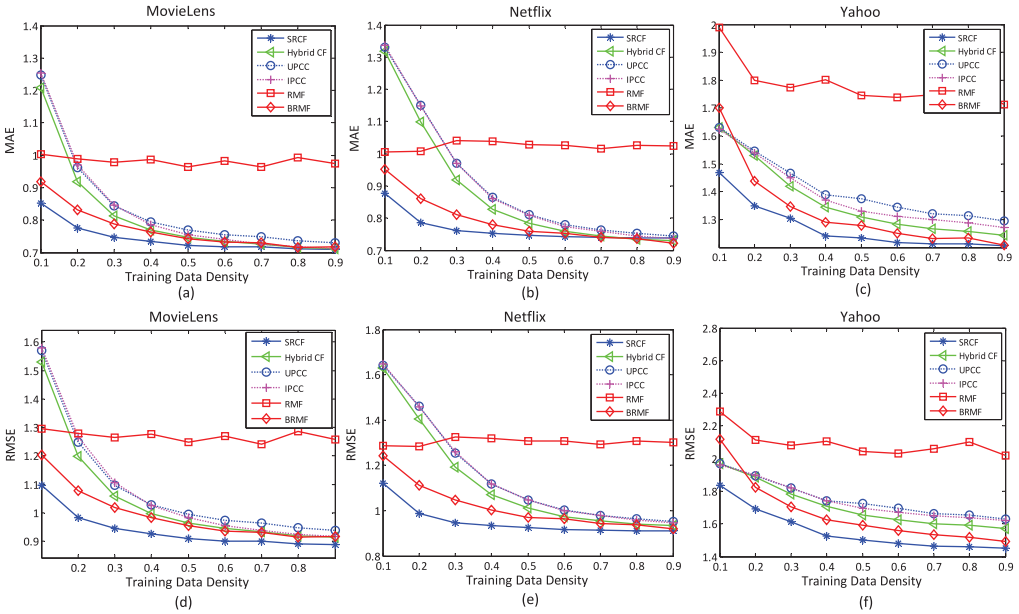

Fig. 4. MAE and RMSE comparison between our approach and traditional CF algorithms.

In the three experimental datasets, users and items without any rating information were filtered out. After that, about 70% of the data of each dataset were selected as training data, and the remaining 30% were used as test data. Given a training set, in order to simulate different sparsity levels, we randomly removed some of its entries to make the training data density vary from 10% to 90%, with a step value of 10%. Finally, we applied different methods to the prepared training sets, and evaluated the prediction accuracy of different methods on the corresponding test datasets. The parameter α is set to 0.5, 0.5, and 0.3 and λ is set to 0.6, 0.5, and 0.7 for the three datasets, respectively, because these values yield the optimal performance. We will investigate the impacts of the two parameters in Sections 5.4 and 5.5.

The comparison of prediction accuracy between different methods is illustrated in Figure 4. Figures 4(a) to (c) report the MAE comparison of different methods on MovieLens 1M, Netflix, and Yahoo! Music datasets, respectively. Figures 4(d) to (f) show the RMSE comparison of different methods on the three datasets. We can see that the proposed similarity reinforcement technique improves the rating prediction accuracy significantly compared with the traditional CF algorithms, especially when the training data become sparser. That testifies the excellent capability of our approach to handle data sparsity.

The boxplots in Figure 5 show another observation that the variation of prediction errors yielded by the proposed approach is significantly lower than those by other methods. For example, on the MovieLens 1M dataset, 50% of the absolute prediction errors are within (0.31, 1.12) with a range = 0.81 for the proposed approach, within (0.36, 1.43) with a range = 1.07 for RMF, within (0.32, 1.18) with a range = 0.86 for BRMF, within (0.36, 1.40) with a range = 1.04 for the hybrid CF, within (0.37, 1.48) with a range = 1.11 for UPCC, and within (0.37, 1.49) with a range = 1.12 for IPCC. It is similar on the other two datasets. The results indicate that the prediction accuracy of the proposed approach is more stable than that of traditional CF algorithms across different datasets.

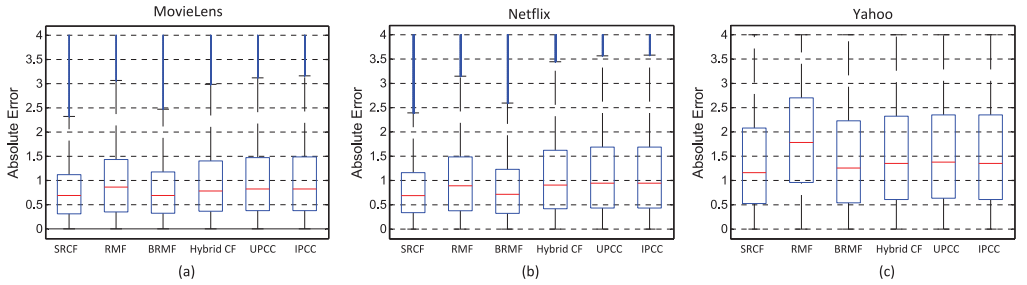

Fig. 5. Boxplots of absolute prediction errors of different methods.

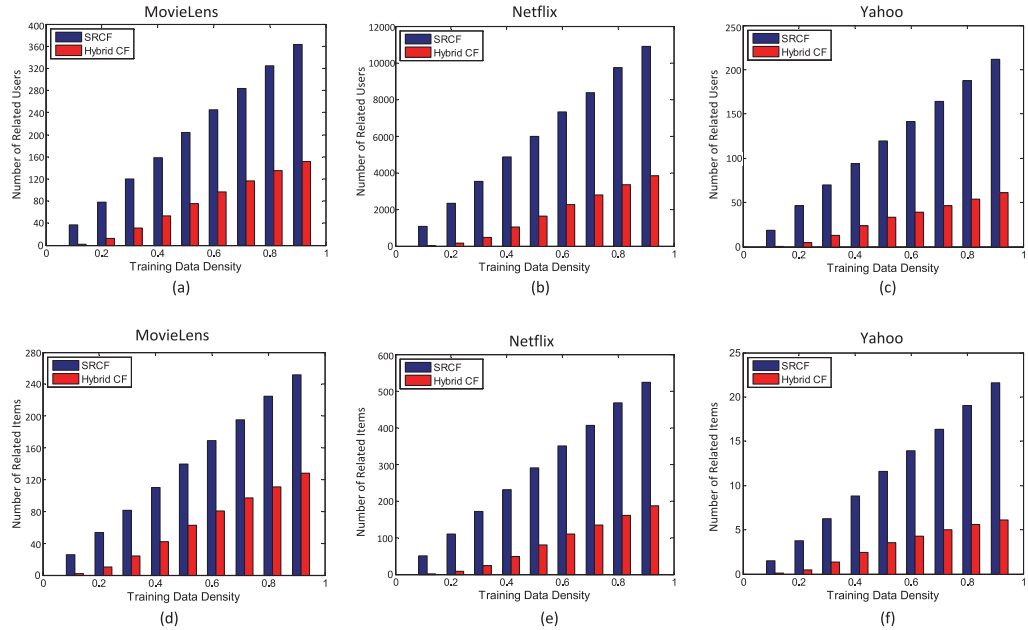

Fig. 6. Average number of related users and items.

Furthermore, we recorded the average numbers of related users and related items discovered by different approaches under different training data densities and report the results in Figure 6 (Given an active user and a candidate item, the similar neighbors of the active user who also rated the candidate item are called *related users*. The similar neighbors of the candidate item that were also rated by the active user are called *related items*.) We can see that our approach can discover more related users and items than the hybrid CF can. Since UPCC and IPCC detected the same number of related users and related items, respectively, as the hybrid CF did, we do not present the results of the two methods. In conclusion, our approach can find more related users and items using the comprehensive similarity reinforcement mechanism, meaning that our approach can extract more historical information from related neighbors, which helps to achieve higher prediction accuracy.

5.4. Impact of α

The damping factor α is used to control the degree that the reinforced similarity relies on its previous value or the contribution of the reinforcement procedure. In this section,

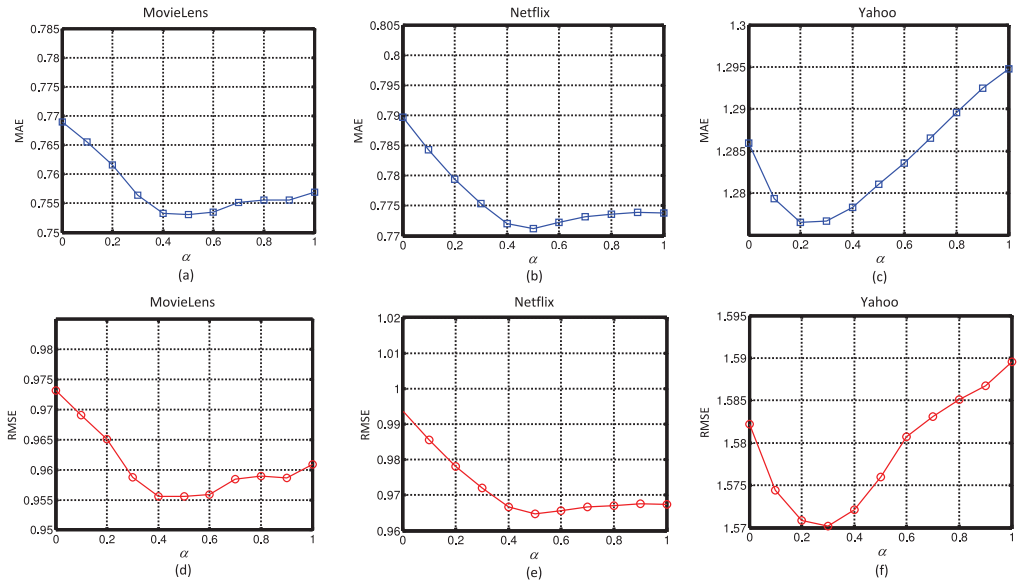

Fig. 7. Impact of α.

in order to investigate the impact of different values of α, we set another adjustable parameter $\lambda = 0.5$ and varied α from 0 to 1 with a step value of 0.1 and investigated how MAE and RMSE change with α under three different training data densities, 0.3, 0.6, and 0.9. Figures 7(a) to (c) illustrate the relationship between the average MAE under three densities and α. Figures 7(d) to (f) show the relationship between the average RMSE and α. We can see from Figure 7 that the α value has impacts on the prediction accuracy. The optimal values of α on the three datasets are 0.5, 0.5, and 0.3.

5.5. Impact of λ

The parameter λ provides our prediction with the flexibility to vary the dependence on the user-based and item-based prediction results. If $\lambda = 1$, we only consider information from similar users; if $\lambda = 0$, we only consider historical information from similar items; if $0 < \lambda < 1$, we fuse information from both similar users and similar items to predict missing ratings for active users.

To study the impact of the parameter λ on the performance of our CF method, we set another adjustable parameter α equal to 0.5, 0.5, and 0.3 on the three datasets, respectively, and varied λ from 0 to 1 with a step value of 0.1 and investigated the prediction accuracy of our algorithm under three different training data densities, 0.3, 0.6, and 0.9. Figures 8(a) to (c) show the relationship between the average MAE under three densities and λ, and Figures 8(d) to (f) show the relationship between the average RMSE and λ.

In Figure 8, we observe that the value of λ impacts the recommendation results, which calls for a suitable λ value for higher prediction accuracy. The optimal λ values are 0.6, 0.5, and 0.7 on the three datasets, respectively.

6. COMPUTATIONAL COMPLEXITY DISCUSSION

In this section, we compare the computational complexity of our approach with that of MF approaches.

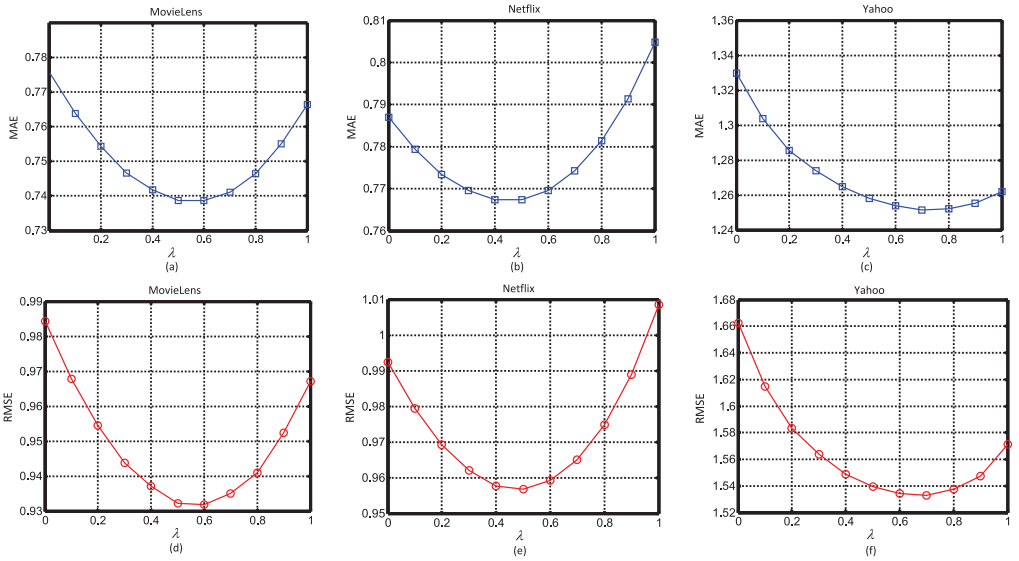

Fig. 8. Impact of λ.

Computational Complexity of SRCF. There are three main procedures in our SRCF approach, the PCC similarity computing, the similarity reinforcement, and the rating prediction. We analyze the computational complexities of the three procedures:

PCC similarity computing. Suppose that $|\bar{I}_u|$ ($|\bar{I}_u| \ll N$) is the average number of items that each user has rated and $|\bar{U}_i|$ ($|\bar{U}_i| \ll M$) is the average number of users that each item has interacted with. The time complexity of user similarity computation based on Equations (1) and (3) is $O(|\bar{I}_u|)$, since there are at most $|\bar{I}_u|$ co-rated items between two users. Similarly, the time complexity of item similarity computation based on Equations (2) and (4) is $O(|\bar{U}_i|)$, since there are at most $O(|\bar{U}_i|)$ common users between two items. Based on that, the time complexity of the similarity computation for a total of M^2 pairs of users is $O(M^2|\bar{I}_u|)$. Similarly, the time complexity of the similarity computation for a total of N^2 pairs of items is $O(N^2|\bar{U}_i|)$.

Similarity reinforcement. After similarity computing, user and item similarity reinforcements are performed using Equations (5) and (8). When performing similarity reinforcement between each pair of users, the similarities between a total of $|\bar{I}_u|^2$ pairs of items should be examined. Suppose that the maximum number of iterations is T. The time complexity for user similarity reinforcement is $O(TM^2|\bar{I}_u|^2)$. Similarly, the time complexity for item similarity reinforcement is $O(TN^2|\bar{U}_i|^2)$.

Rating prediction. The time complexity of the user-based prediction for a missing rating value is $O(M)$, since at most M similar users are used for prediction. Similarly, the time complexity of the item-based prediction for a missing rating value is $O(N)$, since at most N similar items are used for prediction. The hybrid prediction is a linear combination of the user-based and item-based predictions, so its time complexity is $O(M + N)$.

Computational Complexity of MF. In comparison, MF approaches map both users and items to a joint latent factor space of dimensionality f ($f \ll M, N$), and user-item interactions are modeled as inner products in that space. Accordingly, each user u is associated with a vector $\boldsymbol{p}_u \in \mathbb{R}^f$ and each item i is associated with a vector $\boldsymbol{q}_i \in \mathbb{R}^f$. MF approaches mainly include two procedures: model training and rating prediction.

Model training. In the model training procedure, *stochastic gradient descent* is a popular method. Given a training case ($userId$, $itemId$, $rating$), the algorithm predicts the rating value, computes the prediction error, and updates the corresponding user vector and item vector in the opposite direction of the gradient. Suppose that the maximum number of iteration is T_1. The time complexity of the model training procedure is $O(2T_1Kf)$, where K is the number of training cases, which is approximately equal to $M|\bar{I}_u|$ or $N|\bar{U}_i|$.

Rating prediction. The rating prediction procedure uses the dot product of a user vector and an item vector to denote the user's overall interest in this item. Therefore, the time complexity of predicting a rating value for an active user on a candidate item is $O(f)$, where f is the dimensionality of the latent factor space.

Based on the above analysis, the time complexity of the similarity computation and reinforcement of our SRCF approach is $O(M^2|\bar{I}_u| + N^2|\bar{U}_i| + TM^2|\bar{I}_u|^2 + TN^2|\bar{U}_i|^2)$. Because f is much smaller than M and N, and K is approximately equal to $M|\bar{I}_u|$ or $N|\bar{U}_i|$, the time complexity of the similarity computing and reinforcement of SRCF is higher than the time complexity ($O(2T_1Kf)$) of the model training of MF approaches, even though the iteration number T in our approach is usually smaller than T_1 in MF approaches. Additionally, the time complexity of the rating prediction of our approach is $O(M + N)$, which is also higher than that of MF approaches ($O(f)$). Moreover, our algorithm needs to store an $M \times M$ user similarity matrix and an $N \times N$ item similarity matrix, while MF approaches only need to store M f-dimensional user vectors and N f-dimensional item vectors ($f \ll M,\ N$), so the storage requirement of our approach is higher than that of MF. Both the execution time and storage requirements of the two kinds of approaches are determined by their algorithm mechanisms. In the future, we will optimize our algorithm as well as explore the possibility to accelerate it by means of parallel computing.

7. CONCLUSIONS

In this article, we proposed a novel similarity reinforcement mechanism to augment traditional memory-based CF approaches for tackling the data sparsity problem. The main advantage of our approach is that it does not rely on any auxiliary data source. It makes better use of known but limited user-item rating information to reveal more implicit similarity relationships between users or items. The more similar neighbors discovered by our algorithm, the more abundant historical data are provided, and the more reliable and accurate missing rating predictions can be made. Additionally, our approach provides an integrated framework for the user and item similarity reinforcements that allows them to enhance each other. Experimental results show that our approach can achieve higher prediction accuracy than traditional memory-based and model-based CF methods. However, the work presented in this article still has some limitations. Our approach cannot handle cold-start users or items that do not have any rating information. Meanwhile, how to guarantee good performance in terms of execution time and storage when applying our algorithm to very large datasets is still a problem. In the future, we will optimize the algorithm as well as explore the possibility to accelerate it by means of parallel computing.

REFERENCES

[1] Hyung Jun Ahn. 2008. A new similarity mmeasure for collaborative filtering to alleviate the new user cold-starting problem. *Information Sciences* 178, 1 (2008), 37–51.

[2] Yoav Bergner, Stefan Droschler, Gerd Kortemeyer, Saif Rayyan, Daniel Seaton, and David E. Pritchard. 2012. Model-based collaborative filtering analysis of student response data: Machine-learning item response theory. In *Proceedings of the 5th International Conference on Educational Data Mining*. 95–102.

[3] John S Breese, David Heckerman, and Carl Kadie. 1998. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann, 43–52.

[4] John Canny. 2002. Collaborative filtering with privacy via factor analysis. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 238–245.

[5] Oscar Celma. 2010. *Music Recommendation*. Springer.

[6] Wei-Sheng Chin, Yong Zhuang, Yu-Chin Juan, and Chih-Jen Lin. 2015. A fast parallel stochastic gradient method for matrix factorization in shared memory systems. *ACM Transactions on Intelligent Systems and Technology (TIST)* 6, 1 (2015), 2.

[7] Mukund Deshpande and George Karypis. 2004. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)* 22, 1 (2004), 143–177.

[8] Sarik Ghazarian and Mohammad Ali Nematbakhsh. 2015. Enhancing memory-based collaborative filtering for group recommender systems. *Expert Systems with Applications* 42, 7 (2015), 3801–3812.

[9] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. 1992. Using collaborative filtering to weave an information tapestry. *Communications of the ACM* 35, 12 (1992), 61–70.

[10] Miha Grčar, Dunja Mladenič, Blaž Fortuna, and Marko Grobelnik. 2005. *Data Sparsity Issues in the Collaborative Filtering Framework*. Springer.

[11] Guibing Guo, Jie Zhang, and Daniel Thalmann. 2012. A simple but effective method to incorporate trusted neighbors in recommender systems. In *User Modeling, Adaptation, and Personalization*. Springer, 114–125.

[12] Guibing Guo, Jie Zhang, and Daniel Thalmann. 2014. Merging trust in collaborative filtering to alleviate data sparsity and cold start. *Knowledge-Based Systems* 57 (2014), 57–68.

[13] KANG Hanhoon and Seong Joon Yoo. 2007. SVM and collaborative filtering-based prediction of user preference for digital fashion recommendation systems. *IEICE Transactions on Information and Systems* 90, 12 (2007), 2100–2103.

[14] Charif Haydar, Anne Boyer, and Azim Roussanaly. 2012. Hybridising collaborative filtering and trust-aware recommender systems. In *Proceedings of the 8th International Conference on Web Information Systems and Technologies*.

[15] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. 2004. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)* 22, 1 (2004), 5–53.

[16] Antonio Hernando, Jesús Bobadilla, and Fernando Ortega. 2016. A non negative matrix factorization for collaborative filtering recommender systems based on a Bayesian probabilistic model. *Knowledge-Based Systems* 97 (2016), 188–202.

[17] Thomas Hofmann. 2004. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems (TOIS)* 22, 1 (2004), 89–115.

[18] Mohsen Jamali and Martin Ester. 2009. Trustwalker: A random walk model for combining trust-based and item-based recommendation. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 397–406.

[19] Mohsen Jamali and Martin Ester. 2010. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the 4th ACM Conference on Recommender Systems*. ACM, 135–142.

[20] Wenjun Jiang, Jie Wu, and Guojun Wang. 2015. On selecting recommenders for trust evaluation in online social networks. *ACM Transactions on Internet Technology (TOIT)* 15, 4 (2015), 14.

[21] Rong Jin, Joyce Y Chai, and Luo Si. 2004. An automatic weighting scheme for collaborative filtering. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* 337–344.

[22] Liping Jing, Peng Wang, and Liu Yang. 2015. Sparse probabilistic matrix factorization by laplace distribution for collaborative filtering. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI'15)*. 25–31.

[23] Hamza Kaya and Ferda Nur Alpaslan. 2010. Using social networks to solve data sparsity problem in one-class collaborative filtering. In *Proceedings of the 7th International Conference on Information Technology: New Generations (ITNG)*. IEEE, 249–252.

[24] Ioannis Konstas, Vassilios Stathopoulos, and Joemon M. Jose. 2009. On social networks and collaborative recommendation. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 195–202.

[25] Yehuda Koren, Robert Bell, Chris Volinsky, and others. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[26] George Lekakos and Petros Caravelas. 2008. A hybrid approach for movie recommendation. *Multimedia Tools and Applications* 36, 1–2 (2008), 55–70.

[27] Haifeng Liu, Zheng Hu, Ahmad Mian, Hui Tian, and Xuzhen Zhu. 2014. A New User Similarity Model to Improve the Accuracy of Collaborative Filtering. *Knowledge-Based Systems* 56, 12 (2014), 156–166.

[28] Xin Luo, Yunni Xia, and Qingsheng Zhu. 2012. Incremental collaborative filtering recommender based on regularized matrix factorization. *Knowledge-Based Systems* 27 (2012), 271–280.

[29] Hao Ma, Irwin King, and Michael R. Lyu. 2007. Effective missing data prediction for collaborative filtering. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 39–46.

[30] Hao Ma, Irwin King, and Michael R. Lyu. 2009. Learning to Recommend with Social Trust Ensemble. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 203–210.

[31] Augusto Q. Macedo, Leandro B. Marinho, and Rodrygo L. T. Santos. 2015. Context-aware event recommendation in event-based social networks. In *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, 123–130.

[32] Yashar Moshfeghi, Benjamin Piwowarski, and Joemon M. Jose. 2011. Handling data sparsity in collaborative filtering using emotion and semantic based features. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 625–634.

[33] Weike Pan, Evan Wei Xiang, Nathan Nan Liu, and Qiang Yang. 2010. Transfer Learning in Collaborative Filtering for Sparsity Reduction. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, Vol. 10. 230–235.

[34] Xochilt Ramirez-Garcia and Mario García-Valdez. 2014. Post-filtering for a restaurant context-aware recommender system. In *Recent Advances on Hybrid Approaches for Designing Intelligent Systems*. Springer, 695–707.

[35] Steffen Rendle. 2012. Factorization machines with libFM. *ACM Transactions on Intelligent Systems and Technology (TIST)* 3, 3 (2012), 57.

[36] Paul Resnick and Hal R. Varian. 1997. Recommender systems. *Commun. ACM* 40, 3 (1997), 56–58.

[37] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2011. *Introduction to Recommender Systems Handbook*. Springer.

[38] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*. ACM, 285–295.

[39] Marin Silic, Goran Delac, and Sinisa Srbljic. 2015. Prediction of atomic web services reliability for Qos-aware recommendation. *IEEE Transactions on Services Computing* 8, 3 (2015), 425–438.

[40] Sanjeevan Sivapalan, Alireza Sadeghian, Hossein Rahnama, and Asad M. Madni. 2014. Recommender systems in e-commerce. In *Proceedings of the 2014 World Automation Congress (WAC)*. IEEE, 179–184.

[41] Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. 2009. Scalable collaborative filtering approaches for large recommender systems. *Journal of Machine Learning Research* 10, Mar (2009), 623–656.

[42] Mingdong Tang, Wei Liang, Buqing Cao, and Xiangyun Lin. 2015. Predicting quality of cloud services for selection. *International Journal of Grid and Distributed Computing* 8, 4 (2015), 257–268.

[43] Chih-Fong Tsai and Chihli Hung. 2012. Cluster ensembles in collaborative filtering recommendation. *Applied Soft Computing* 12, 4 (2012), 1417–1425.

[44] Qingqing Tu and Le Dong. 2010. An intelligent personalized fashion recommendation system. In *Proceedings of the 2010 International Conference on Communications, Circuits and Systems (ICCCAS)*. IEEE, 479–485.

[45] Paula Cristina Vaz, David Martins de Matos, Bruno Martins, and Pavel Calado. 2012. Improving a hybrid literary book recommendation system through author ranking. In *Proceedings of the 12th ACM/IEEE-CS Joint Conference on Digital Libraries*. ACM, 387–388.

[46] Slobodan Vucetic and Zoran Obradovic. 2005. Collaborative filtering using a regression-based approach. *Knowledge and Information Systems* 7, 1 (2005), 1–22.

[47] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative Deep Learning for Recommender Systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1235–1244.

[48] Jun Wang, Arjen P. De Vries, and Marcel J. T. Reinders. 2006. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 501–508.

[49] Jian Wu, Liang Chen, Yipeng Feng, Zibin Zheng, Meng Chu Zhou, and Zhaohui Wu. 2013. Predicting quality of service for selection by neighborhood-based collaborative filtering. *IEEE Transactions on Systems, Man, and Cybernetics* 43, 2 (2013), 428–439.

[50] Meng-Lun Wu, Chia-Hui Chang, and Rui-Zhe Liu. 2014. Integrating content-based filtering with collaborative filtering using co-clustering with augmented matrices. *Expert Systems with Applications* 41, 6 (2014), 2754–2761.

[51] Yao Wu, Christopher DuBois, Alice X. Zheng, and Martin Ester. 2016a. Collaborative denoising auto-encoders for top-n recommender systems. In *Proceedings of the 9th ACM International Conference on Web Search and Data Mining*. ACM, 153–162.

[52] Yao Wu, Xudong Liu, Min Xie, Martin Ester, and Qing Yang. 2016b. CCCF: Improving collaborative filtering via scalable user-item co-clustering. In *Proceedings of the 9th ACM International Conference on Web Search and Data Mining*. ACM, 73–82.

[53] Hongzhi Yin, Yizhou Sun, Bin Cui, Zhiting Hu, and Ling Chen. 2013. LCARS: A location-content-aware recommender system. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 221–229.

[54] Quan Yuan, Shiwan Zhao, Li Chen, Yan Liu, Shengchao Ding, Xiatian Zhang, and Wentao Zheng. 2009. Augmenting collaborative recommender by fusing explicit social relationships. In *Proceedings of the Workshop on Recommender Systems and the Social Web (Recsys'09)*.

[55] Yongfeng Zhang, Min Zhang, Yiqun Liu, Shaoping Ma, and Shi Feng. 2013. Localized matrix factorization for recommendation based on matrix block diagonal forms. In *Proceedings of the 22nd International Conference on World Wide Web*. ACM, 1511–1520.

[56] Zibin Zheng, Hao Ma, Michael R. Lyu, and Irwin King. 2009. WSREC: A collaborative filtering based Web service recommender system. In *Proceedings of the IEEE 7th International Conference on Web Services*. 437–444.

[57] Zibin Zheng, Hao Ma, Michael R. Lyu, and Irwin King. 2011. QoS-aware Web service recommendation by collaborative filtering. *IEEE Transactions on Services Computing* 4, 2 (2011), 140–152.

Received April 2016; revised February 2017; accepted March 2017