Applied Intelligence
https://doi.org/10.1007/s10489-018-1289-9

Stochastic trust network enriched by similarity relations to enhance trust-aware recommendations

Mina Ghavipour[1] · Mohammad Reza Meybodi[1]

Abstract

Collaborative filtering (CF) is the most popular recommendation approach that has been extensively employed in recommender systems. However, it suffers from some weaknesses, including problems with cold start users, data sparsity and difficulty in detecting malicious users. Trust-based recommender systems can overcome these weaknesses by using the ratings of trusted users. However, since users often provide few trust statements, trust networks are typically sparse and therefore the cold start and sparsity problems still remain. In this paper, we use the positive correlation between trust and interest similarity to enrich trust network by similarity relations and propose a stochastic trust propagation-based method, called LTRS, which utilizes the enriched trust network to provide enhanced recommendations. In comparison with existing recommender systems combining trust and similarity information, the proposed system (1) incorporates both trust and similarity relations in the trust propagation process and, in this way, increases the coverage and accuracy of predictions; and (2) addresses the dynamic nature of both trust and similarity by modelling the enriched network as a stochastic graph, and continuously captures their variations during the recommendation process and not at fixed intervals. The experimental results indicate that the proposed method can significantly improve the recommendation accuracy and coverage.

Keywords Collaborative filtering · Trust · Interest similarity · Trust propagation · Stochastic network · Learning automata

1 Introduction

Due to the incredible growth of information on the World Wide Web in the recent years, searching and finding contents, products or services that may be of interest for users has become a very difficult task. Recommender systems (RSs) help overcome the information overload problem by studying the preferences of online users and suggesting items they might like. Many companies and Web sites have implemented these systems to recommend products/information/services to their users in a more accurate manner, therefore improving the company's profits.

Collaborative filtering (CF) [1] is a representative recommendation technique that has been widely used in recommender systems. CF operates on the assumption that the active user will prefer those items which his similar users prefer. It generates a prediction for a given item by aggregating the past ratings of a set of suitable users with similar preferences. CF-based recommender system has been extensively studied in the literature, and many approaches have been proposed for it [2–6]. Despite the significant success of collaborative filtering technique, it has been known to reveal two major problems: data sparsity [2, 44] and cold start [4–6, 45]. While there exists the huge number of items available, users normally rate only few of them. Therefore, the number of items rated in common between each pair of users is not enough for similarity measures to accurately measure user similarities. In addition, new users cannot receive any reliable recommendations, since they have not yet provided any rating information in the system.

In order to overcome the above-mentioned problems, researchers have proposed to use trust information rather than similarity between users in CF. The intuition is that in real life users rely more on recommendations from people they trust [7]. However, in the recommendation context, trust must reflect the user similarity to a certain extent in order to have meaningful results [8]. Previous studies have shown that incorporating trust networks into

✉ Mina Ghavipour
mina_ghavipour@aut.ac.ir

1 Department of Computer Engineering and IT, Amirkabir University of Technology, Tehran, Iran

Published online: 31 August 2018

recommender systems improves the quality of predictions and recommendations [9–11].

Many social applications, such as Filmtrust.com and Epinions.com, provide a web of trust to allow users to express their trust on others. Unfortunately, the web of trust in these applications is typically sparse since only a few of users specify their trust relationships, and most of them tend to provide no trust statements. Therefore, the data sparsity and cold start problems still remain. A number of research works have attempted to address this issue by making recommendations based on both similarity and trust information [12–18]. The common idea in these works is to predict the rating of an active user on a target item by using the ratings from his similar users in addition to those from directly/indirectly trusted users. It has been shown that there exists a strong correlation between trust and user similarity when the trust network is tightly bound to a particular application [19]. As a result, similarity is used as an additional measure in determining the value of implicit trust between users in trust management systems [20–26]. The assumption is that similar users are most likely to trust each other. Based on this, in this paper we use the similarity relations between users to enrich trust network and mitigate the sparsity problem of this network. We then propose a recommender system, called LTRS, which produces predictions for an active user by propagating trust through the enriched trust network and aggregating ratings from directly/indirectly and explicitly/implicitly trusted users. In comparison with existing works, the proposed system incorporates both similarity and trust relations in the trust propagation process and, in this way, increases the coverage and accuracy of predictions.

Since users' interests vary with time, the similarity and the intensity of trust between them also change continuously as time passes. Therefore, the enriched trust network can be considered as a stochastic graph with continuous time-varying edge weights. Only few research works on recommender systems have taken into account the dynamic nature of similarity and (or) trust [12, 27]. However, even these works either focus only on the temporal changes of similarity [27], or despite considering the dynamicity of both similarity and trust, update their information at fixed time intervals and not during the recommendation process [12]. To address this issue, we use the stochastic trust propagation algorithm Dytrust proposed in our previous work [28] for propagating trust along reliable paths in the enriched trust network. This algorithm utilizes learning automata (LA) to dynamically capture the temporal changes of edge weights during the propagation process and update the found reliable paths based on these variations.

In order to validate the efficiency of the proposed recommender system LTRS, we conduct comprehensive experiments on the well-known dataset Epinions. The experimental results indicate that the proposed algorithm effectively improves both the accuracy and coverage of recommendations.

This paper consists of the following. Section 2 provides an overview of the related research on CF-based and trust-based CF recommender systems. A brief background of learning automata is presented in Section 3. In Section 4, we describe the proposed recommender system based on stochastic trust propagation. The experimental results and discussion are conducted in Section 5. Finally, Section 6 summarizes and concludes our present work.

2 Related work

Recommender systems (RSs) have advanced in the ability to filter out unnecessary information and present the most relevant data to users. These systems make use of different information sources for providing users with recommendations of items. They attempt to balance factors like accuracy, diversity and novelty in their recommendations. Recommender systems can be generally categorized into content-based filtering (CB) [29–32] and collaborative filtering (CF) [33–39]. Content-based filtering recommends new items based on their similarity to the items already rated by the user. On the other hand, in collaborative filtering approach the rating of the user for a new item is predicted based on past ratings of similar users. Collaborative filtering techniques play an important role in designing recommendation systems.

2.1 CF-based recommender systems

Collaborative filtering is based on the way in which humans make their decisions in real life: besides on our personal experiences, we also base our decisions on the experiences of our acquaintances. In this technique, users are allowed to give ratings about a set of items (e.g. books, movies, musics, etc.) in such a way that when enough rating data is stored on the system, recommendations can be made to each user based on information provided by those users who have the most in common with him.

CF technique has been most widely used in recommender systems. Generally, there exist two main approaches to CF: memory-based and model-based CF techniques [40, 41]. Memory-based techniques [42–45] use similarity measures to predict the preference of a user for new or unrated items based on user-item ratings stored in the system. These techniques are conceptually simple and easily implementable. They also produce good quality recommendations. In contrast, model-based techniques use rating data to learn a predictive model. These approaches present better scalability under large datasets in comparison with memory-based ones. However, they require expensive model-building processes, and have a trade-off between predication performance and scalability. Among the widely used models, we have Bayesian classifiers [46], neural networks [47], fuzzy systems [48], genetic algorithms [2], latent features [3] and matrix factorization [4, 49].

Despite the significant success of collaborative filtering technique, it suffers from some problems, including data sparsity, cold start and malicious users. Since users typically rate only few of the millions of items, the rating matrix usually has the high level of sparsity [50]. Therefore, similarity measures used by CF-based recommender systems often encounter processing problems (from insufficient mutual ratings for computing user similarity). The cold start problem [33, 51–53] refers to the situation where a new user just enters the RS. The user cannot receive any personalized recommendations based on CF technique, since he has not yet provided any rating in the system. Moreover, CF-based recommender systems can experience shilling attacks [54, 55], in which many positive ratings are generated for a product, while the products from competitors receive negative rating. Standard CF techniques are highly vulnerable to such attacks [56].

In order to overcome the above-mentioned problems, researchers have proposed to incorporate trust networks into recommender systems. Using trust information in these systems can also improve the quality of recommendations [9–11].

2.2 Trust-based CF recommender systems

Trust is an important area of research in recommender systems [57]. Users connected by a web of trust significantly exhibit higher similarity on items than non-connected users [58]. Therefore, social influences can play a more important role than the similarity of past ratings [59, 60]. Previous studies have shown that trust information not only improves the recommendation performance in terms of the prediction accuracy and coverage via a trust propagation approach, but also mitigates some problems inherent in CF based recommender systems [61, 62]. With trust information, the data sparsity problem can be alleviated since it is no longer necessary to measure the rating similarity for finding like-minded users. In addition, for a cold start user with no past ratings, recommender systems still can make good recommendations using the preferences of his trusted neighbourhood. Moreover, since recommendations are based only on ratings provided by trusted users, it is possible to resist malicious users who are trying to influence the recommendation accuracy.

Trust-based CF techniques adopt one of these two main approaches: incorporating trust as a replacement for the user similarity, or using trust in combination with the user similarity. Recommender systems based on the first approach [9, 62, 63] recommend new items to a user from his trusted users. Since only a few users specify their trust relations and most of them tend to provide no trust statements, trust networks are typically sparse. As a result, the data sparsity and cold start problems still remain in these systems. For this reason, researchers attempted to take into account both similarity and trust information for making high quality recommendations [12–18]. For instance, Yan et al. [12] developed a novel recommendation method, called CITG, based on a two-faceted web of trust. In their proposed method, a web of trust derived by implicit trust relations, called as interest similarity graph (ISG), is constructed for an active user by measuring interest similarities between the user and the others. Then, another web of trust derived by explicit trust relations is formed by computing trust values between the active user and other directly/indirectly connected users in the trust network. The resulting web of trust is called as directed trust graph (DTG). Finally, ISG and DTG are combined to generate a two-faceted web of trust which mitigates the sparsity and cold start problems. Authors in [13] proposed a multi-view clustering method which clusters users from the views of both user similarities and trust relations. Their intuition was that a cluster with few users fails to produce reliable rating predictions for a given item. They showed that the proposed method improves both the recommendation accuracy and coverage. Work in [14] presented a trust-aware collaborative filtering method based on reliability, called RTCF. In this method, an initial trust network for an active user is constructed by combination of the trust statements and the similarity values. Using the trust network, the initial rating of an unseen item is predicted for the active user. After that, a trust based reliability measure is proposed to evaluate the quality of the predicted rating and based on this evaluation, the trust network is reconstructed by removing useless users with a reliability value lower than a predefined threshold. Finally, the new trust network is used to predict the final rate of the unseen item. In order to alleviate the rating sparsity problem, Mao et al. [15] considered different user relations (such as the rating similarity and trust) in a multigraph and developed a multigraph ranking model to identify the nearest neighbours of an active user for the recommendation purpose. Authors also proposed a random walk-based social network propagation model which is applied to every single-relational social network to enrich the original data of the network before constructing the multigraph. At last, the user's closest neighbours are used to make the CF rating predictions for unseen items. Gohari et al. [25] proposed a new confidence-based recommendation (CBR) approach which employs four different confidence models and derives users' and items' confidence values from both local and global perspectives. In this approach, the neighbourhood formation process for a user relies on both implicit trust values between users and their interpretations of ratings. CBR predicts an active user's rating of the target item based on the most confident neighbours of the user.

Another challenge in recommender systems that must be addressed is the time-dependent nature of similarity and trust. User interests may vary over time which in turn change the rating similarities between users. Trust also is dynamic and changes as time passes and users continue social interactions or observations such as rating common items [64]. Only few research works have considered the dynamicity of similarity and trust in their proposed RSs [12, 27]. Bedi and Sharma [27] presented a trust-based ant recommender system (TARS). This method creates an implicit trust network for each user based on user-item rating matrix and provides recommendations for the active user by continuously updating implicit trust between users and selecting the best neighbourhood using ant colony metaphor. Authors in [12] considered the temporal nature of both trust and similarity by dynamically updating their proposed two-faceted web of trust. In the ISG, the interest intensity of all the edges is updated by new item ratings, while in the DTG, the trust intensity for all the edges is updated at fixed intervals by referral feedback ratings.

Some research works suggested that implicit trust relations can be generated from users' rating information [65–69]. Commonly, these works assumed that users will trust others who have similar preferences consistently. Based on this, in this paper we construct an enriched trust network consisting of implicit and explicit trust relations to mitigate the sparsity problem of trust networks. We then propose a stochastic trust propagation-based recommender system, called LTRS, that exploits the enriched trust network to provide high quality recommendations. Considering the time-dependent nature of trust and similarity, LTRS uses the learning automata-based algorithm proposed in our previous work on stochastic trust propagation [28] to more efficiently discover reliable trust paths and, at the same time, capture the temporal changes of implicit and explicit trust during the propagation process. The proposed system LTRS differs from the above-mentioned works in a number of ways. First, in contrast to methods such as TidalTrust [70], TARS [27], and CBR [25], our algorithm LTRS exploits both user similarities and explicit trust relations for making recommendations to mitigate the sparsity presented by rating information and web of trust. Second, unlike existing models in the literature, LTRS propagates trust through an enriched network consisting of both implicit and explicit trust relations and, in this way, improves the coverage and accuracy of rating predictions. Finally, in comparison with methods such as TARS [27], and CITG [12], LTRS addresses the dynamic nature of both trust and similarity, and continuously captures their temporal variations during the recommendation process and not at fixed intervals. Since users' interests vary with time, it is highly probable that the similarity and the intensity of trust between them change during the recommendation process and therefore the item ratings predicted by recommender systems may become less relevant because of these variations. Using learning automata, LTRS not only continuously capture the temporal changes of trust and similarity, but also accelerates the propagation process.

3 Learning automata

Learning automata (LA) [71, 72] is a reinforcement learning approach for adaptive decision making in unknown random environments. LA attempts to learn the optimal action from a finite set of allowable actions using repetitive interactions with the random environment and, in this way, improve its performance. At each time step t, the automaton chooses an action $\alpha(t)$ from its allowable actions based on the corresponding action probability distribution, and applies $\alpha(t)$ to the random environment. The environment evaluates the chosen action $\alpha(t)$ and responses in turn with a reinforcement signal $\beta(t)$ (either a reward or a penalty) with a certain probability. At last, the action probability distribution of the automaton is updated based on the signal $\beta(t)$ received from the environment. By repeating these interactions, the automaton learns the optimal action which is the action with the minimum penalty probability. The interaction between an automaton and its random environment is depicted in Fig. 1.

Stochastic LA can be classified into two main categories: variable structure learning automata (VSLA) and fixed structure learning automata (FSLA). VSLA is defined by a quadruple $\langle \alpha, \beta, p, T \rangle$, where $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ denotes the set of actions which the automaton chooses from, $\beta = \{\beta_1, \beta_2, \ldots, \beta_k\}$ is the set of input signals to the automaton, $p = \{p_1, p_2, \ldots, p_r\}$ denotes the action probability vector with p_i indicating the selection probability of action α_i, and T is the learning algorithm that updates the action probability vector of the automaton

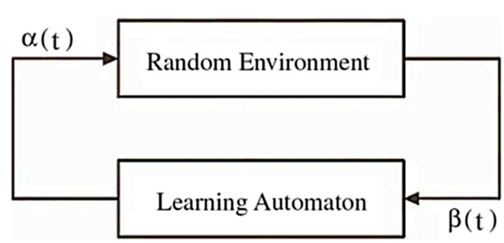

Fig. 1 The interaction between a learning automaton and its random environment

in terms of the random environment's response, i.e. $p(t+1) = T[\alpha(t), \beta(t), p(t)]$, where the inputs are the chosen action $\alpha(t)$, the environment's response $\beta(t)$ and the probability vector $p(t)$ at time step t.

Let $\alpha_i(t)$ be the action chosen by the automaton at time step t. The action probability vector $p(t)$ kept over the action set is updated as given in (1), if $\alpha_i(t)$ is rewarded by the random environment, and otherwise $p(t)$ is updated according to (2).

$$p_j(t+1) = \begin{cases} p_j(t) + a\left[1 - p_j(t)\right] & j = i \\ (1-a)\, p_j(t) & \forall_j \neq i \end{cases} \tag{1}$$

$$p_j(t+1) = \begin{cases} (1-b)\, p_j(t) & j = i \\ \left(\frac{b}{r-1}\right) + (1-b)\, p_j(t) & \forall j \neq i \end{cases} \tag{2}$$

where a and b respectively denote reward and penalty parameters which determine the amount of increases and decreases in the action probabilities, and r is the number of available actions for the automaton. If $a = b$, the learning algorithm T is a linear reward–penalty (L_{R-P}) algorithm, if $a \gg b$, T is a linear reward–ϵ penalty ($L_{R-\epsilon P}$) algorithm, and if $b = 0$, T is a linear reward–Inaction (L_{R-I}) algorithm in which the probability vector $p(t)$ remains unchanged when the chosen action is penalized by the random environment. Learning automata has already found many applications in the literature, for example, graph sampling [73, 74], fuzzy membership function optimization [75], trust propagation [28, 76].

3.1 Variable action set learning automata

Variable action set learning automaton is an automaton for which the number of available actions varies over time. The procedure of choosing an action and updating the corresponding action probability vector for such a learning automaton can be described as follows. At each time step t, a subset $\hat{\alpha}(t)$ of all the allowable actions (i.e. $\hat{\alpha}(t) \subseteq \alpha$) is available for the automaton to choose from. The elements of $\hat{\alpha}(t)$ are chosen randomly by an external factor. Let $K(t) = \sum_{\alpha_i \in \hat{\alpha}(t)} p_i(t)$ denote the sum of the probabilities of all the available actions in $\hat{\alpha}(t)$. The scaled selection probability of each action $\alpha_i \in \hat{\alpha}(t)$ is computed as

$$\hat{p}_i(t) = \frac{p_i(t)}{K(t)} \qquad \forall \alpha_i \in \hat{\alpha}(t). \tag{3}$$

The automaton randomly chooses one of the available actions in $\hat{\alpha}(t)$ based on its scaled action probability vector $\hat{p}(t)$. Depending on the reinforcement signal received from the random environment, the probability vector $\hat{p}(t)$ of the automaton is updated. Finally, $\hat{p}(t)$ is rescaled according to (4).

$$p_i(t+1) = \hat{p}_i(t+1)\, K(t) \qquad \forall \alpha_i \in \hat{\alpha}(t) \tag{4}$$

3.2 Distributed learning automata

Distributed learning automata (DLA) [77] is defined as a network of interconnected learning automata which work together to solve a particular problem. Formally, a DLA can be described by a quadruple $\langle A, E, L, A_0 \rangle$, where $A = \{A_1 A_2, \ldots, A_n\}$ denotes a set of automata, $E \subset A \times A$ is an edge set (such that edge e_{ij} refers to the action α_{ij} of the automaton A_i), L presents a set of learning algorithms for updating the action probability vectors of the automata, and A_0 refers to the root automaton from which the activation process of the learning automata begins. Each time A_0 is activated, it randomly chooses one of its actions (i.e. outgoing edges) according to the action probability vector. As a result of the action selection, the automaton on the other end of the chosen edge is activated. The activated automaton also chooses an action at random which in turn activates another automaton. The activation process of the automata is repeated until a leaf automaton (i.e. an automaton that directly interacts with the random environment) is reached. The chosen actions along the path induced by the activation process serve as the inputs to the random environment. Depending on the signal received from the environment, the activated automata along the path update their corresponding action probability vectors. Figure 2 depicts an example of DLA.

4 Proposed recommender system

In this section, we first introduce basic notations used for describing our proposed recommender system.

Typically in a recommender system, there exists a set of users $U = \{u_1 u_2, \cdots, u_N\}$ and a set of items $I = \{i_1 i_2, \cdots, i_M\}$. Each user u_i specifies his preferences by rating a subset I_i of items by some values. The rating of a user u_i on an item i_k is denoted by $r_{i,k}$. The task of a recommender system is as follows. Given an active user $u_s \in U$ and a target item $i_d \in I$ such that $i_d \notin I_s$ (i.e. $r_{s,d}$ is unknown). The recommender system predicts the rating of u_s on i_s, denoted by $\hat{r}_{s,d}$, based on the existing ratings.

In the following, we describe the proposed recommender system LTRS based on stochastic trust propagation in

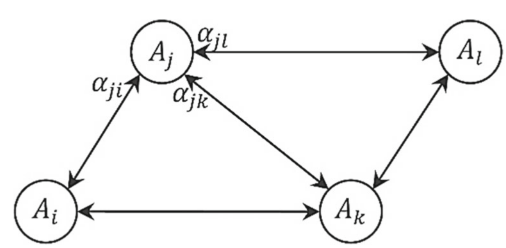

Fig. 2 Distributed learning automaton

two phases: constructing an enriched trust network and generating predictions for the active user.

4.1 Constructing an enriched trust network

A trust network is defined as a weighted digraph $G^T(UE^TT)$, where $U = \{u_1u_2, \cdots, u_N\}$ represents the set of N nodes which corresponds to users, $E^T \subseteq U \times U = \{e^T_{ij}|u_iu_j \in U\}$ is the set of M directed edges or trust relations that connect users, and $T = \{t_{ij}|e^T_{ij} \in E^T\}$ denotes the set of edge weights or trust values between users. If user u_i trusts user u_j, then there is a directed trust relation $e^T_{ij} \in E^T$ from u_i to u_j with weight $t_{ij} \in T$ referring to the value of this trust. Since trust is dynamic, G^T is modelled as a stochastic graph with trust weights being random variables. We assume that each trust weight t_{ij} takes real values in range [0, 1], with 0 referring to no trust and 1 to full trust.

Although explicit trust-based recommender systems are characterized by high prediction coverage and accuracy, but trust networks are typically sparse and this matter affects the quality of recommendations. In order to alleviate this problem, we enrich the trust network G^T by adding implicit trust relations among users. Considering the strong correlation between trust and similarity, implicit trust between two users can be computed using their interest similarity. That is, two notions implicit trust and similarity refer to the same concept and so might be used interchangeably. Based on this, we compute interest similarities between each pair of users and construct a similarity network which in combination with the trust network G^T mitigates the sparsity of both networks.

Let a similarity (implicit trust) network be a weighted digraph $G^S(UE^SS)$ with the user set U, the set of similarity relations $E^S \subseteq U \times U = \{e^S_{ij}|u_iu_j \in U\}$, and the set of similarity weights $S = \{s_{ij}|e^S_{ij} \in E^S\}$, such that if two users u_i and u_j have some items commonly rated with positive correlation, then there exists two directed similarity relations $e^S_{ij}e^S_{ji} \in E^S$ in both directions between them respectively with weights $s_{ij}s_{ji} \in S$ being equal to the value of interest similarity $sim(ij)$. Considering the dynamic nature of similarity, G^S is also a stochastic graph in which similarity weights are random variables. We use the Pearson correlation coefficient for computing the similarity between two users u_i and u_j in terms of the items rated in common by them at the current time.

$$corr\,(i, j) = \frac{\sum_{i_k \in I_{i,j}} (r_{i,k} - \bar{r}_i)(r_{j,k} - \bar{r}_j)}{\sqrt{\sum_{i_k \in I_{i,j}} (r_{i,k} - \bar{r}_i)^2}\sqrt{\sum_{i_k \in I_{i,j}} (r_{j,k} - \bar{r}_j)^2}} \tag{5}$$

where $r_{i,k}$ is the rating of user u_i for item i_k, $I_{i,j} = I_i \cap I_j$ is the set of items that two users rated in common, and $\bar{r}_i$ and $\bar{r}_j$ denote the average of ratings given by u_i and u_j, respectively. The value of $corr\,(i, j)$ is in the range $[-1, 1]$. Since zero and negative correlations indicate that the ratings expressed by two users are uncorrelated or correlated in opposite directions, they are not useful for our purpose and no edges will be added for them to the similarity network G^S.

The Pearson correlation coefficient measures the extent to which two users u_i and u_j with similar preferences linearly relate with each other. However, it does not determine the degree of confidence the user u_i should have in u_j and vice versa. We consider the similarity confidence as a sigmoid function of the number of items commonly rated by u_i and u_j [78] and compute the interest similarity between these two users as follows.

$$sim\,(i, j) = \frac{1}{1 + e^{-\frac{|I_{i,j}|}{2}}} \times corr(ij) \tag{6}$$

where $corr\,(i, j) > 0$ and $|I_{i,j}|$ denotes the size of the set $I_{i,j}$. Using the sigmoid function, we avoid favouring the size of $I_{i,j}$ too much and keep the similarity value $sim\,(i, j)$ in the range [0, 1]. Once the rating similarities between all pairs of users were computed, for each pair with positive correlation two similarity relations in both directions will be added to G^S and, in this way, the similarity network G^S is constructed. We then combine both networks of similarity and trust to generate an enriched trust network $G(UEW)$, where U is the user set, $E = E^S \cup E^T$ denotes the set of implicit and explicit trust relations such that there is a directed relation $e_{ij} \in E$ from user u_i to user u_j if u_i trusts u_j in G^T (i.e. $e^T_{ij} \in E^T$), or u_i and u_j are positively correlated in G^S (i.e. $e^S_{ij} \in E^S$), or both, $W = \{w_{ij}|e_{ij} \in E\}$ is the set of integrated edge weights such that each weight w_{ij} is a random variable whose value equals $com_t(ij)$, a combination of similarity and explicit trust from u_i to u_j at time t that is computed as

$$com_t(ij) = \begin{cases} \frac{2\sigma_{ij}(t) \times \tau_{ij}(t)}{\sigma_{ij}(t) + \tau_{ij}(t)} & if\, \sigma_{ij}\,(t) \neq 0\; and\; \tau_{ij}\,(t) \neq 0 \\ \tau_{ij}\,(t) & else\, if\; \sigma_{ij}\,(t) = 0\; and\; \tau_{ij}\,(t) \neq 0 \\ \sigma_{ij}\,(t) & elseif\, \sigma_{ij}\,(t) \neq 0\; and\; \tau_{ij}\,(t) = 0 \\ 0 & else \end{cases} \tag{7}$$

where $\sigma_{ij}(t)$ $(\tau_{ij}(t))$ is the expected value of implicit (explicit) trust from user u_i to user u_j which is computed as given in (8) and (9). The value of $com_t(ij)$ lies in the interval [0, 1]. The advantage of using the harmonic mean is that it is robust to large differences among its inputs, so that high values will be obtained only when both expected similarity and trust values are high. The harmonic mean has been widely used in the literature [12, 13, 27, 57] to integrate similarity and trust.

$$\sigma_{ij}(t) = \frac{\sum_{l=1}^{|\vartheta_{ij}(t)|} \lambda^{|\vartheta_{ij}(t)|-l} \vartheta_{ij}^{l}(t)}{\sum_{l=1}^{|\vartheta_{ij}(t)|} \lambda^{|\vartheta_{ij}(t)|-l}} \tag{8}$$

$$\tau_{ij}(t) = \frac{\sum_{l=1}^{|\omega_{ij}(t)|} \lambda^{|\omega_{ij}(t)|-l} \omega_{ij}^{l}(t)}{\sum_{l=1}^{|\omega_{ij}(t)|} \lambda^{|\omega_{ij}(t)|-l}} \tag{9}$$

In the above equations, $\vartheta_{ij}(t)$ ($\omega_{ij}(t)$) denotes the set of similarity (trust) weights observed on the relation e_{ij}^{S} (e_{ij}^{T}) until the current time and $\vartheta_{ij}^{l}(t)$ ($\omega_{ij}^{l}(t)$) refers to lth member of this set. Since more recent observations should be given relatively greater weight, we use the decay factor $\lambda \in [0, 1]$ to control the rate at which old similarity and trust weights are discounted.

4.2 Generating predictions for active user

In order to predict the rating of an active user u_s on a target item i_d, we temporarily add a node i_d to the enriched trust graph and connect each user u_i who has already rated the item i_d to the new node by a directed edge e_{id} with the weight $r_{i,d}$ referring to the rating value. In this way, the rating prediction problem can be converted to a trust inference problem and solved by propagating trust along reliable paths from u_s to i_d. Figure 3 illustrates different steps of our proposed system LTRS using a simple example. LTRS uses the stochastic trust propagation algorithm Dytrust proposed in our previous work [28] which exploits learning automata to more efficiently discover reliable trust paths and, at the same time, capture the temporal changes of edge weights during the propagation process. The stochastic enriched trust network $G(UEW)$, the active user u_s and the indirectly connected target i_d form the inputs to DyTrust, and its output is the predicted rating $\hat{r}_{s,d}$. Let N_d be the set of users rating the target item i_d, which are referred to as the direct neighbours of i_d. Using the Dytrust algorithm, at first the most reliable trust path to each user $u_v \in N_d$ is discovered with respect to samples taken from edge weights, and the strength of the found path is considered as the reliability of the direct neighbour u_v. Then, the final rating $\hat{r}_{s,d}$ on i_d for the active user u_s is predicted by aggregating the ratings of the direct neighbours weighted by their reliability values, as given in (10) [76].

$$\hat{r}_{s,d} = \bar{r}_s + \frac{\sum_{u_v \in N_d} R_v \left(r_{v,d} - \bar{r}_v\right)}{|N_d|\, Max_w} \tag{10}$$

where R_v refers to the reliability value of direct neighbour u_v, $\bar{r}_v$ is the average of ratings provided by u_v, and Max_w denotes the maximum implicit/explicit trust weight which is equal to 1 in this paper.

In order to estimate the reliability of direct neighbours, Dytrust first constructs a distributed learning automata (DLA) isomorphic to the input graph G in such a way, each node u_i is equipped with a learning automaton A_i. The action set α_i of A_i contains all the neighbours implicitly or explicitly trusted by u_i, namely the size of α_i equals to the number of u_i's outgoing relations. The action probability vector of each automaton is updated based on the learning algorithm L_{R-I}.

After that, for each direct neighbour $u_v \in N_d$ the Dytrust algorithm performs the following two tasks:

1. Initializing the action probability vectors

   The (11) is used to initialize the action probability vector p_i of each automaton A_i. This equation gives higher selection probability to neighbours who are highly trusted.

$$p_{ij}(t) = \frac{com_{t-1}(i, j)}{\sum_{u_k \in \alpha_i} com_{t-1}(i, k)} \forall u_j \in \alpha_i \tag{11}$$

   where $p_{ij}(t)$ denotes the probability of selecting node u_j by A_i at time t and $com_{t-1}(ij)$ is the integrated value of similarity and trust from v_i to v_j at time $t-1$ and is computed according to (7).

2. Learning the most reliable path to direct neighbour

   DyTrust attempts to discover the most reliable trust path to the direct neighbour u_v and estimate the reliability of u_v based on the strength of the found path. For this purpose, it repeats the following three subtasks until the stopping criteria are reached.

   a. Discovering a trust path

      In this subtask, the aim is to find a trust path π to u_v by a series of automaton activations starting from the root automaton A_s corresponding to the source u_s. Each activated automaton A_i determines the next hop along the path π. The set of available actions for A_i contains all trusted neighbours of v_i except those whose corresponding automata have been already activated along π. A_i selects one of its available actions, say action v_j, based on the scaled action probability vector. As a result of the action selection, a sample is taken from each of relations e_{ij}^{S} and e_{ij}^{T}, the value of $com_t(ij)$ is updated according to (7) and the automaton A_j on the other end of the relation e_{ij} is activated. If A_j belongs to the set N_d and the minimum of integrated weights along the current path π, called the path strength R_π, is larger or equal to the maximum strength R_j already obtained for v_j, namely if $A_j \in N_d$ and $R_\pi = (com_t(i, j)) \geq R_j$, then $R_j = R_\pi$. The activation process is repeated until the neighbour u_v is reached or the activated automaton has no available action.

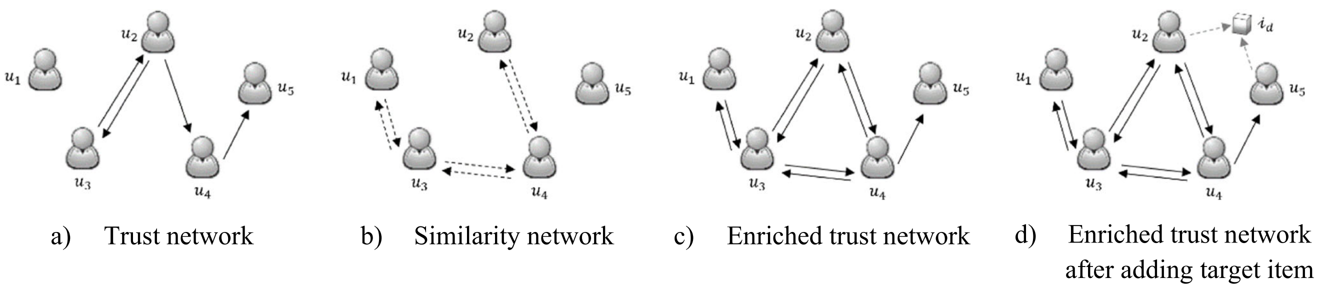

Fig. 3 Description of how to construct an enriched trust network: **a** Trust network, where any directed edge indicates who trusts whom, **b** Similarity network, where there are two directed edges in both direction between any two positively correlated users, **c** Enriched trust network, where there is a directed edge from u_i to u_j if u_i trusts u_j or u_i is positively correlated to u_j or both, **d** To predict the rating of target item i_k, a node i_k is temporarily added to the enriched trust network, and the users u_2 and u_5, who have already rated i_k, are connected to the new node, each by a temporary directed edge

b. Evaluating the found path

If the found path π ends in the direct neighbour u_v and $R_\pi \geq R_v$, the learning automata activated along π receive a reward. For each automaton A_j, the value of the reward parameter a at time t is given as [77].

$$a_j(t) = \frac{a_i(t)}{p_{ij}(t+1)} \forall e_{ij} \in \pi \tag{12}$$

where $p_{ij}(t+1)$ denotes the selection probability of action u_j by A_i after rewarding the automaton.

c. Checking stop criteria

Two previous subtasks are repeated until the path probability, namely the product of the probability of choosing relations along the path π, is greater than a certain threshold P or the number of found paths exceeds a predefined threshold K. In this situation, the maximum strength R_v is considered as the reliability of the rating $r_{v,d}$ provided by u_v for the target i_d.

5 Experiments

In order to evaluate the performance of the proposed recommender system LTRS, we conduct empirical experiments on the Epinions dataset and compare our algorithm with the pure Collaborative Filtering (CF) [44], TidalTrust [70], TARS [27], CITG [12], RTCF [14] and CBR [25]. These methods have been commonly used in the literature for evaluating the performance of proposed recommender systems [12, 14, 25, 78, 79]. In all the experiments, these settings are considered for the parameters in the LTRS algorithm: the path probability threshold P is set to 0.9, the threshold K for the number of traversed paths to 10000 and the decay factor λ to 0.9. Experimental results are averaged over 10 independent runs.

5.1 Experimental settings

5.1.1 Dataset

The experiments are conducted on a version of the Epinions dataset (www.epinions.com) published by the authors in [80]. In the Epinions website, users are able to review items and express their opinions about them by assigning numeric ratings in the range of 1 to 5. Moreover, users can also indicate others as trustworthy. The extracted dataset contains 916,149 item ratings from 22,164 users who have rated at least once among 296,277 different items. The trust network of the Epinions dataset also consists of 18,098 users, with 15,892 users trusted by at least one other user and 15,451 users trusting at least one other user, with a total of 355,727 trust statements. We use the RandomWalk sampling algorithm to produce a smaller subgraph from the trust network dataset. For a sampling fraction of 0.1, the sampled subgraph contains 1,809 users with 22,115 trust connections among them. We also extract the rating data for these users, which includes 63,897 items with 104,715 ratings expressed for them. Finally, using the technique proposed in Section 4.1, an initial similarity network is constructed between users.

In the experiments, we need similarity and trust networks to be stochastic graphs with real-valued edge weights varying over time. For this purpose, we use the technique proposed by Richardson et al. [81], which has been commonly used in the literature [76, 82–84]. This technique assigns to each user u_i a quality value $q_i \in [0, 1]$ showing the probability that a statement issued by u_i is true. In this paper, we consider the average of similarity between a user u_i and his direct neighbours in the similarity network as the quality q_i of u_i. Since similarity weights are between 0 and 1, it is ensured that quality values are also in the interval [0, 1]. After that, the implicit/explicit trust weight from user u_i to u_j is assumed to be a random variable with a continuous uniform distribution on the interval

$\left[\max\left(q_j - \delta_{ij}, 0\right), min(q_j + \delta_{ij}, 1)\right]$, where $\delta_{ij} = \frac{1-q_i}{2}$ is a noise parameter which determines how accurate u_i is at estimating the quality of u_j that he is implicitly/explicitly trusting.

5.1.2 Evaluation technique

Typically, the leave-one-out method [85] is used to simulate a dynamic recommendation process. At every test round, one item rating is taken out from the dataset and the compared algorithms attempt to predict it using the trust network and the remaining ratings. The quality of the various algorithms is measured by comparing the predicted rating and the actual rating. We repeat this process through the entire rating dataset, and then average the results.

5.1.3 Evaluation measures

The performance is measured in terms of rating coverage, predictive accuracy and classification accuracy. The rating coverage (RC) refers to the fraction of ratings for which a RS algorithm is able to produce a predicted rating. Mean absolute error (MAE) and root mean square error (RMSE) are popular predictive accuracy metrics to measure the closeness of rating predictions relative to the true ratings:

$$MAE = \frac{\sum_{u_i \in U} \sum_{i_k \in I} \left|\hat{r}_{i,k} - r_{i,k}\right|}{n} \tag{13}$$

$$RMSE=\sqrt{\frac{\sum_{u_i \in U} \sum_{i_k \in I} (\hat{r}_{i,k} - r_{i,k})^2}{n}} \tag{14}$$

where n denotes the total number of ratings, $r_{i,k}$ and $\hat{r}_{i,k}$ respectively refer to the actual rating and the predicted rating for user u_i on item i_k. In general, lower MAE and RMSE values indicate higher prediction accuracy.

To measure classification accuracy, we use the metrics precision and recall which are the most popular metrics for evaluating the accuracy of RS algorithms in making decision. Precision (Pr) measures the ability of a system to suggest item that is truly relevant for an active user and is computed as

$$Pr = \frac{I_A \cap I_B}{I_B} \tag{15}$$

where I_A denotes the total number of relevant items and I_B is the total number of items recommended to the user. Recall (Re) measures the ability of a system to gather the relevant content to the active user and is computed as the fraction of items are actually relevant and are successfully recommended, as given below.

$$Re = \frac{I_A \cap I_B}{I_A} \tag{16}$$

The metrics precision and recall are clearly conflicting in nature. If the number of recommended items increases, then the value of precision is decreased, while at the same time recall increases. One may combine them by using the F_1 measure which is the harmonic mean of precision and recall. A High F_1 corresponds to balanced combination between recall and precision.

$$F_1 = \frac{2 \times Pr \times Re}{Pr + Re} \tag{17}$$

5.2 Experimental results

5.2.1 Comparison with other methods

The first experiment is conducted to compare the predictive performance of our proposed method LTRS with CF, TidalTrust, TARS, CITG, RTCF and CBR methods in terms of accuracy and coverage on the Epinions dataset. The comparison results are shown in Table 1. As one can see from this table, our proposed method LTRS outperforms all the other methods under six metrics.

Epinions has a very sparse rating data and a relatively small number of trust statements. Since the traditional CF considers only the rating information in making recommendations, it shows the worst predictive and classification accuracy with the lowest coverage in comparison with the other methods. By using explicit trust relations instead of rating similarities, TidalTrust improves the rating coverage and obtains higher accuracy than CF. TARS performs better than two previous methods because it creates implicit trust network based on the rating information, thereby reducing the sparsity of user similarity. The method CITG mitigates the sparsity problem of rating data and trust network by incorporating both information into the recommendation process. In this way, CITG increases the coverage and produces more accurate predictions than those generated by TARS. RTCF also uses the combination of similarity values and trust statements. Since this method removes the users with lower reliability values from the trust network, it has a lower rating coverage than CITG. Although CITG shows better Re and F_1 results, RTCF achieves higher prediction accuracy compared to CITG.

CBR does not consider explicit trust relations between users and computes implicit trust based on the rating information. It gives close results to CITG in terms of MAE and RMSE measures and close results to TARS in terms of the coverage measure. However, the classification accuracy of CBR is higher than that of the previous methods. Similar to CITG and RTCF, LTRS combines similarity and trust networks. However, by propagating trust along paths consisting of both types of relations, the proposed method LTRS achieves the advantage of higher coverage

Table 1 Comparison between different methods in terms of accuracy and coverage on the Epinions dataset

Method	Metric					
	Predictive accuracy		Classification accuracy			RC
	MAE	RMSE	Pr	Re	F_1	
CF	0.9427	1.3327	0.7916	0.8376	0.8140	0.3776
TidalTrust	0.9127	1.2509	0.7910	0.8712	0.8292	0.4301
TARS	0.8990	1.2132	0.7913	0.8920	0.8386	0.4578
CITG	0.8711	1.1591	0.7988	0.9251	0.8573	0.4863
RTCF	0.8563	1.1103	0.8015	0.9043	0.8498	0.4722
CBR	0.8779	1.1624	0.7997	0.9633	0.8739	0.4591
LTRS	0.8339	1.0862	0.8016	0.9724	0.8788	0.5179

and accuracy compared to the other methods. Considering the temporal variation of both similarity and trust during the recommendation process also improves the prediction performance of LTRS.

5.2.2 Performance for cold-start users

This experiment aims to study the effectiveness of the proposed method LTRS in dealing with new users (cold-start users) who have provided only a few or even no ratings. For this purpose, we consider only users who rated less than 5 items in the Epinions dataset and report the accuracy and coverage of compared methods in Table 2. According to the results of this table, LTRS has the best performance in handling cold-start users as comparing to the other methods.

The results prove that using trust information in the recommendation process can effectively alleviate the cold-start problem. In comparison with CF, the method TidalTrust shows higher prediction and classification accuracy and can provides reliable recommendations for a more number of new users. TARS mitigates the cold-start user problem by considering popular users with high reputation as trusted friends of new users. In this way, it achieves better results than TidalTrust. The combination of trust and similarity information and using default recommenders in CITG makes this method more powerful in predicting missing ratings. CITG performs better than TARS in terms of prediction coverage and classification accuracy. However, the MAE and RMSE values of CITG are slightly worse than those of TARS.

The RTCF method shows higher predictive accuracy, but lower F_1 and coverage in comparison with CITG. By making predictions based on the opinions of neighbours with high global reputations, CBR provides reasonable recommendations for new users. This method has the highest F_1 value compared to the previous methods. The results for LTRS also show that trust propagation along both similarity and trust relations increases the coverage and accuracy of our proposed method. LTRS significantly outperforms the other methods under all the metrics.

5.2.3 Performance for sparse rating data

In this experiment, we examine the impact of different rating sparsity levels on the performance of our method LTRS as compared to the other methods. This experiment is conducted under four different protocols, keeping 100%, 75%, and so on down to 25% of the total number of ratings

Table 2 Performance of different methods in handling the cold-start problem on the Epinions dataset

Method	Metric					
	Predictive accuracy		Classification accuracy			RC
	MAE	RMSE	Pr	Re	F_1	
CF	1.0473	1.4309	0.8226	0.7727	0.7969	0.3058
TidalTrust	0.9801	1.3500	0.8284	0.7872	0.8073	0.5670
TARS	0.9511	1.2736	0.8361	0.8245	0.8303	0.6135
CITG	0.9742	1.3167	0.8447	0.8915	0.8675	0.6911
RTCF	0.9386	1.2515	0.8510	0.8666	0.8587	0.6317
CBR	0.9723	1.3089	0.8417	0.9153	0.8770	0.6179
LTRS	0.9279	1.2185	0.8693	0.9760	0.9196	0.7423

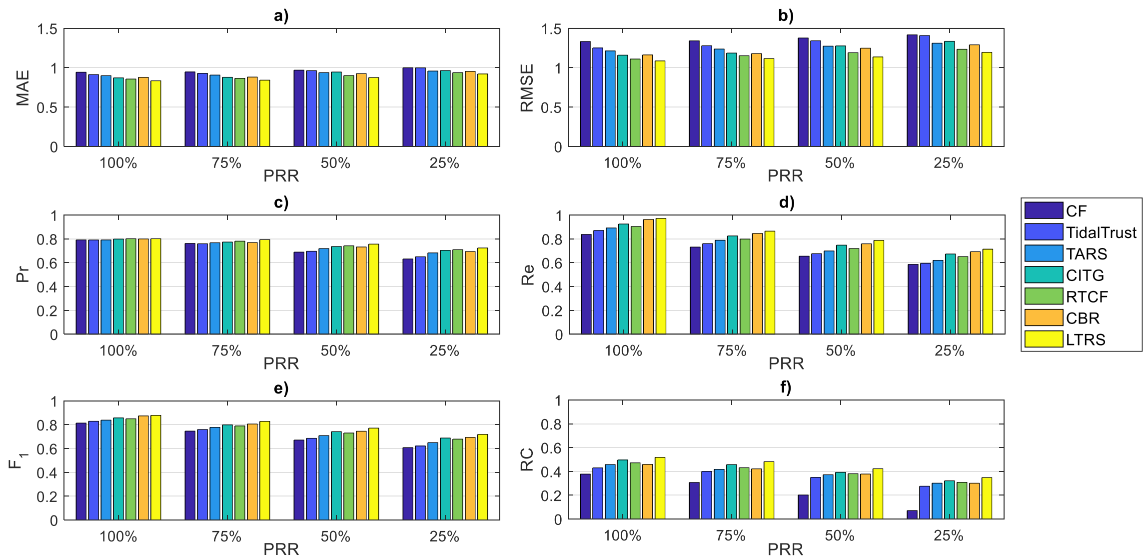

Fig. 4 Impact of different rating sparsity levels on the performance of different methods

and discarding the rest in the Epinions dataset. Figure 4 shows the results of compared methods for different levels of sparsity. In this figure, PRR represents the percentage of retained ratings.

From these figures, we can see that the performance of all methods decreases with increasing rating sparsity. However, those methods that benefit from trust information in the recommendation process can effectively handle the rating sparsity problem. Among them, our proposed method LTRS always shows the best performance. According to the results of Fig. 4a–e, the increase of sparsity level decreases the predictive and classification accuracy for all the methods. With respect to the RC metric (Fig. 4(f)), the negative effect of the sparsity problem on the rating coverage is much stronger for CF compared to the other methods. Generally, LTRS performs better than the others in dealing with the rating sparsity problem.

6 Conclusion

In this paper, we proposed a stochastic trust propagation-based recommender system called LTRS. In our proposed system, we mitigate the sparsity problem of trust network by constructing an enriched trust network which consists of both implicit and explicit trust relations. LTRS predicts the rating of an active user on a target item by propagating trust through the enriched trust network. To address the dynamic nature of both similarity and trust, LTRS uses a stochastic trust propagation algorithm based on learning automata which dynamically captures the temporal changes of implicit/explicit trust weights during the propagation process and updates the found reliable paths based on these variations.

The experimental results on the well-known dataset Epinions demonstrated that the proposed system LTRS can improve both the accuracy and coverage of recommendations in comparison with its competitors. The results also confirmed that LTRS can effectively handle the issues of rating data sparsity and cold-start users.

Acknowledgments This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

1. Adomavicius G, Tuzhilin A (2005) Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Trans Knowl Data Eng 17:734–749
2. Gao L, Li C (2008) Hybrid personalized recommended model based on genetic algorithm. In: 4th international conference on wireless communications, networking and mobile computing, 2008. WiCOM'08. IEEE, pp 1–4
3. Zhong J, Li X (2010) Unified collaborative filtering model based on combination of latent features. Expert Syst Appl 37:5666–5672
4. Luo X, Xia Y, Zhu Q (2012) Incremental collaborative filtering recommender based on regularized matrix factorization. Knowledge-Based Syst 27:271–280
5. Wang Y, Deng J, Gao J, Zhang P (2017) A hybrid user similarity model for collaborative filtering. Inf Sci (Ny) 418:102–118
6. Ren L, Wang W (2018) An SVM-based collaborative filtering approach for Top-N web services recommendation. Futur Gener Comput Syst 78:531–543
7. Sinha RR, Swearingen K (2001) Comparing Recommendations Made by Online Systems and Friends. In: DELOS workshop: personalisation and recommender systems in digital libraries

8. Ziegler C-N, Lausen G (2004) Analyzing correlation between trust and user similarity in online communities. In: ITrust. Springer, pp 251–265
9. Massa P, Avesani P (2004) Trust-aware collaborative filtering for recommender systems. CoopIS/DOA/ODBASE (1) 3290:492–508
10. Arazy O, Kumar N, Shapira B (2009) Improving social recommender systems. IT Prof 11
11. Carrer-Neto W, Hernández-Alcaraz ML, Valencia-García R, García-Sánchez F (2012) Social knowledge-based recommender system. Application to the movies domain. Expert Syst Appl 39:10990–11000
12. Yan S, Zheng X, Chen D, Wang Y (2013) Exploiting two-faceted web of trust for enhanced-quality recommendations. Expert Syst Appl 40:7080–7095
13. Guo G, Zhang J, Yorke-Smith N (2015) Leveraging multiviews of trust and similarity to enhance clustering-based recommender systems. Knowledge-Based Syst 74:14–27
14. Moradi P, Ahmadian S (2015) A reliability-based recommendation method to improve trust-aware recommender systems. Expert Syst Appl 42:7386–7398
15. Mao M, Lu J, Zhang G, Zhang J (2017) Multirelational social recommendations via multigraph ranking. IEEE Trans Cybern 47:4049–4061. https://doi.org/10.1109/TCYB.2016.2595620
16. Sheugh L, Alizadeh SH (2018) A novel 2D-Graph clustering method based on trust and similarity measures to enhance accuracy and coverage in recommender systems. Inf Sci (Ny) 432:210–230
17. Deng X, Zhong Y, Lü L et al (2017) A general and effective diffusion-based recommendation scheme on coupled social networks. Inf Sci (Ny) 417:420–434
18. Kalaï A, Zayani CA, Amous I et al (2018) Social collaborative service recommendation approach based on user's trust and domain-specific expertise. Futur Gener Comput Syst 80:355–367
19. Ziegler C-N, Golbeck J (2007) Investigating interactions of trust and interest similarity. Decis Support Syst 43:460–475
20. Bhuiyan T (2013) Trust for intelligent recommendation. Springer, Berlin
21. Golbeck J (2009) Trust and nuanced profile similarity in online social networks. ACM Trans Web 3:12
22. Shambour Q, Lu J (2011) A hybrid trust-enhanced collaborative filtering recommendation approach for personalized government-to-business e-services. Int J Intell Syst 26:814–843
23. Shambour Q, Lu J (2012) A trust-semantic fusion-based recommendation approach for e-business applications. Decis Support Syst 54:768–780
24. Uddin MG, Zulkernine M, Ahamed SI (2008) CAT: a context-aware trust model for open and dynamic systems. In: Proceedings of the 2008 ACM symposium on Applied computing. ACM, pp 2024–2029
25. Gohari FS, Aliee FS, Haghighi H (2018) A new confidence-based recommendation approach: Combining trust and certainty. Inf Sci (Ny) 422:21–50
26. Shambour Q, Lu J (2015) An effective recommender system by unifying user and item trust information for B2B applications. J Comput Syst Sci 81:1110–1126
27. Bedi P, Sharma R (2012) Trust based recommender system using ant colony for trust computation. Expert Syst Appl 39:1183–1190
28. Ghavipour M, Meybodi MR (2018) A dynamic algorithm for stochastic trust propagation in online social networks: Learning automata approach. Comput Commun 123:11–23. https://doi.org/10.1016/j.comcom.2018.04.004
29. Protasiewicz J, Pedrycz W, Kozłowski M et al (2016) A recommender system of reviewers and experts in reviewing problems. Knowledge-Based Syst 106:164–178
30. Lops P, De Gemmis M, Semeraro G (2011) Content-based recommender systems: State of the art and trends. In: Recommender systems handbook. Springer, pp 73–105
31. Martinez-Romo J, Araujo L (2012) Updating broken web links: An automatic recommendation system. Inf Process Manag 48:183–203
32. Pera MS, Ng Y-K (2013) A group recommender for movies based on content similarity and popularity. Inf Process Manag 49:673–687
33. Bobadilla J, Ortega F, Hernando A, Bernal J (2012) A collaborative filtering approach to mitigate the new user cold start problem. Knowledge-Based Syst 26:225–238
34. Bobadilla J, Hernando A, Ortega F, Bernal J (2011) A framework for collaborative filtering recommender systems. Expert Syst Appl 38:14609–14623
35. Bobadilla J, Hernando A, Ortega F, Gutiérrez A (2012) Collaborative filtering based on significances. Inf Sci (Ny) 185:1–17
36. Altingovde IS, Subakan ÖN, Ulusoy Ö (2013) Cluster searching strategies for collaborative recommendation systems. Inf Process Manag 49:688–697
37. Formoso V, FernáNdez D, Cacheda F, Carneiro V (2013) Using profile expansion techniques to alleviate the new user problem. Inf Process Manag 49:659–672
38. Choi IY, Oh MG, Kim JK, Ryu YU (2016) Collaborative filtering with facial expressions for online video recommendation. Int J Inf Manage 36:397–402
39. Wang H, Shao S, Zhou X et al (2016) Preference recommendation for personalized search. Knowledge-Based Syst 100:124–136
40. Ricci F, Rokach L, Shapira B (2011) Introduction to recommender systems handbook. In: Recommender systems handbook. Springer, pp 1–35
41. Su X, Khoshgoftaar TM (2009) A survey of collaborative filtering techniques. Adv Artif Intell 2009:4
42. Linden G, Smith B, York J (2003) Amazon. com recommendations: Item-to-item collaborative filtering. IEEE Internet Comput 7:76–80
43. Lemire D (2005) Scale and translation invariant collaborative filtering systems. Inf Retr Boston 8:129–150
44. Resnick P, Iacovou N, Suchak M et al (1994) GroupLens: an open architecture for collaborative filtering of netnews. In: Proceedings of the 1994 ACM conference on Computer supported cooperative work. ACM, pp 175–186
45. Symeonidis P, Nanopoulos A, Manolopoulos Y (2009) MoviExplain: a recommender system with explanations. In: Proceedings of the third ACM conference on Recommender systems. ACM, pp 317–320
46. Park M-H, Hong J-H, Cho S-B (2007) Location-based recommendation system using bayesian user's preference model in mobile devices. In: International Conference on Ubiquitous Intelligence and Computing. Springer, pp 1130–1139
47. Roh TH, Oh KJ, Han I (2003) The collaborative filtering recommendation based on SOM cluster-indexing CBR. Expert Syst Appl 25:413–423
48. Yager RR (2003) Fuzzy logic methods in recommender systems. Fuzzy Sets Syst 136:133–149
49. Gurini DF, Gasparetti F, Micarelli A, Sansonetti G (2018) Temporal people-to-people recommendation on social networks with sentiment-based matrix factorization. Futur Gener Comput Syst 78:430–439
50. Bobadilla J, Serradilla F (2009) The effect of sparsity on collaborative filtering metrics. In: Proceedings of the 20th Australasian conference on australasian database-volume, vol 92. Australian Computer Society, Inc., pp 9–18

51. Rashid AM, Karypis G, Riedl J (2008) Learning preferences of new users in recommender systems: an information theoretic approach. ACM SIGKDD Explor Newsl 10:90–100
52. Kim H-N, El-Saddik A, Jo G-S (2011) Collaborative error-reflected models for cold-start recommender systems. Decis Support Syst 51:519–531
53. Leung CW, Chan SC, Chung F (2008) An empirical study of a cross-level association rule mining approach to cold-start recommendations. Knowledge-Based Syst 21:515–529
54. Chirita P-A, Nejdl W, Zamfir C (2005) Preventing shilling attacks in online recommender systems. In: Proceedings of the 7th annual ACM international workshop on Web information and data management. ACM, pp 67–74
55. Lam SK, Riedl J (2004) Shilling recommender systems for fun and profit. In: Proceedings of the 13th international conference on World Wide Web. ACM, pp 393–402
56. O'Mahony M, Hurley N, Kushmerick N, Silvestre G (2004) Collaborative recommendation: A robustness analysis. ACM Trans Internet Technol 4:344–377
57. O'Donovan J, Smyth B (2005) Trust in recommender systems. In: Proceedings of the 10th international conference on Intelligent user interfaces. ACM, pp 167–174
58. Lee DH, Brusilovsky P (2009) Does trust influence information similarity? Recomm Syst Soc Web 10
59. Salganik MJ, Dodds PS, Watts DJ (2006) Experimental study of inequality and unpredictability in an artificial cultural market. Science (80-) 311:854–856
60. Bonhard P, Sasse MA (2006) Knowing me, knowing you—Using profiles and social networking to improve recommender systems. BT Technol J 24:84–98
61. He J, Chu WW (2010) A social network-based recommender system (SNRS). In: Data mining for social network data. Springer, pp 47–74
62. Golbeck J (2006) Generating predictive movie recommendations from trust in social networks. In: International Conference on Trust Management. Springer, pp 93–104
63. Avesani P, Massa P, Tiella R (2005) A trust-enhanced recommender system application: Moleskiing. In: Proceedings of the 2005 ACM symposium on Applied computing. ACM, pp 1589–1593
64. Staab S, Bhargava B, Leszek L et al (2004) The pudding of trust: Managing the dynamic nature of trust. IEEE Intell Syst 19:74–88
65. Hwang C-S, Chen Y-P (2007) Using trust in collaborative filtering recommendation. In: International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems. Springer, pp 1052–1060
66. Yuan W, Shu L, Chao H-C et al (2010) ITARS: trust-aware recommender system using implicit trust networks. IET Commun 4:1709–1721
67. Eirinaki M, Louta MD, Varlamis I (2014) A trust-aware system for personalized user recommendations in social networks. IEEE Trans Syst Man Cybern Syst 44:409–421
68. O'Donovan J (2009) Capturing trust in social web applications. In: Computing with social Trust. Springer, pp 213–257
69. Shambour QY (2012) Hybrid recommender systems for personalized government-to-business e-services
70. Golbeck JA (2005) Computing and applying trust in web-based social networks. https://doi.org/10.1017/CBO9781107415324.004
71. Narendra KS, Thathachar MAL (2012) Learning automata: an introduction. Courier Corporation
72. Thathachar MAL, Sastry PS (2011) Networks of learning automata: Techniques for online stochastic optimization. Springer Science & Business Media, Berlin
73. Ghavipour M, Meybodi MR (2017) A streaming sampling algorithm for social activity networks using fixed structure learning automata. Appl Intell. https://doi.org/10.1007/s10489-017-1005-1
74. Ghavipour M, Meybodi MR (2017) Irregular cellular learning automata-based algorithm for sampling social networks. Eng Appl Artif Intell 59:244–259. https://doi.org/10.1016/j.engappai.2017.01.004
75. Ghavipour M, Meybodi MR (2016) An adaptive fuzzy recommender system based on learning automata. Electron Commer Res Appl 20:105–115. https://doi.org/10.1016/j.elerap.2016.10.002
76. Ghavipour M, Meybodi MR (2017) Trust propagation algorithm based on learning automata for inferring local trust in online social networks. Knowledge-Based Syst. https://doi.org/10.1016/j.knosys.2017.06.034
77. Beigy H, Meybodi MR (2006) Utilizing distributed learning automata to solve stochastic shortest path problems. Int J Uncertainty Fuzziness Knowl-Based Syst 14:591–615
78. Jamali M, Ester M (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, pp 397–406
79. Kant V, Bharadwaj KK (2013) Fuzzy computational models of trust and distrust for enhanced recommendations. Int J Intell Syst 28:332–365
80. Tang J, Gao H, Liu H (2012) mTrust: discerning multi-faceted trust in a connected world. In: Proceedings of the fifth ACM international conference on Web search and data mining. ACM, pp 93–102
81. Richardson M, Agrawal R, Domingos P (2003) Trust management for the semantic web. In: International semantic Web conference. Springer, pp 351–368
82. Jiang W, Wu J, Wang G (2015) On selecting recommenders for trust evaluation in online social networks. ACM Trans Internet Technol 15:14
83. Jiang W, Wu J, Li F et al (2016) Trust Evaluation in Online Social Networks Using Generalized Network Flow. IEEE Trans Comput 65:952–963
84. Shekarpour S, Katebi SD (2010) Modeling and evaluation of trust with an extension in semantic web. Web Semant Sci Serv Agents World Wide Web 8:26–36
85. Kohavi R (1995) A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Ijcai. pp 1137–1145

Mina Ghavipour received the M.S. degrees in Computer Engineering from Amirkabir University of Technology in Iran, in 2012. She is currently pursuing the Ph.D. degree in Computer Engineering at Amirkabir University of Technology, Tehran, Iran. Her research interests include social network analysis, network sampling, recommender systems, trust, learning systems, parallel and distributed algorithms, and soft computing.

Mohammad Reza Meybodi received the B.S. and M.S. degrees in Economics from Shahid Beheshti University in Iran, in 1973 and 1977, respectively. He also received the M.S. and Ph.D. degree from Oklahoma University, USA, in 1980 and 1983, respectively in Computer Science. Currently, he is a full professor in Computer Engineering Department, Amirkabir University of Technology, Tehran, Iran. Prior to current position, he worked from 1983 to 1985 as an assistant professor at Western Michigan University, and from 1985 to 1991 as an associate professor at Ohio University, USA. His research interests include wireless networks, fault tolerant systems, learning systems, parallel algorithms, soft computing and software development.