On Deep Learning for Trust-Aware Recommendations in Social Networks

Shuiguang Deng, *Member, IEEE*, Longtao Huang, Guandong Xu, *Member, IEEE*, Xindong Wu, *Fellow IEEE*, and Zhaohui Wu, *Senior Member, IEEE*

***Abstract*— With the emergence of online social networks, the social network-based recommendation approach is popularly used. The major benefit of this approach is the ability of dealing with the problems with cold-start users. In addition to social networks, user trust information also plays an important role to obtain reliable recommendations. Although matrix factorization (MF) becomes dominant in recommender systems, the recommendation largely relies on the initialization of the user and item latent feature vectors. Aiming at addressing these challenges, we develop a novel trust-based approach for recommendation in social networks. In particular, we attempt to leverage deep learning to determinate the initialization in MF for trust-aware social recommendations and to differentiate the community effect in user's trusted friendships. A two-phase recommendation process is proposed to utilize deep learning in initialization and to synthesize the users' interests and their trusted friends' interests together with the impact of community effect for recommendations. We perform extensive experiments on real-world social network data to demonstrate the accuracy and effectiveness of our proposed approach in comparison with other state-of-the-art methods.**

***Index Terms*— Deep learning, recommender systems (RSs), social network, trust.**

I. Introduction

DUE to the explosive growth and influx of information available on the World Wide Web and the rapid introduction of new Web-based services, information overload has become a crucial challenge that users are overwhelmed, and how to help users locating their interested information is becoming a unprecedentedly important task, attracting attention from both research and application domains. To this end, recommender systems (RSs) have emerged as an effective mechanism providing suggestions about what items (e.g., movies, books, music, and so on) users might be potentially interested in [1] and [2]. Such suggestions can facilitate the user experience improvement and the user loyalty to e-commerce Web sites through the accurate recommendations to users. RS has proved in recent years to be a valuable means addressing the above-mentioned information overload problem [3], [4].

Although RSs have widely been studied in both the academia and the industry, some important problems still remain.

1) *Sparsity Problem:* Users usually rate or experience only a small fraction of the available items. As a result, the density of the available ratings in RSs is often less than 1% [5]. Due to this data sparsity, collaborative filtering approaches suffer a lot of difficulties when trying to identify similar users in the system. Consequently, the prediction quality of the RS might be significantly limited.
2) *Cold-Start Problem:* This is a common challenge in RS so far. Here, the cold-start refers to the users, who have expressed no or a few ratings or items which have been rated by no or a small number of users. Due to the lack of sufficient rating data, the similarity-based approaches fail to find out the nearest neighbor users or items and, in turn, deteriorate the recommendation quality via traditional recommendations algorithms.
3) *Trustworthiness Problem:* Traditional RSs usually lack the capability of differentiating users' creditability, i.e., the trustworthiness of users' ratings. In reality, users may trust more on their friends' feedbacks rather than other ordinary users when making decisions. There also exist spam users, who always give fake ratings for malicious purposes. Apparently, such ratings should be excluded in recommendation making.

Therefore, a traditional RS, which purely mines the user-item-rating matrix for recommendations, cannot sufficiently provide accurate and reliable predictions. Considerable research has been conducted to address the above problems, and new solutions have been proposed accordingly. Among these approaches, a kind of efforts is to integrate the side (or complementary) information along with the traditional rating data. In line with this direction, a typical work is called the trust-based RS thanks to the emergence of social networking. The foundation of trusted-based recommendation is the hypothesis that people usually like to refer to their trusted friends' preference to make decisions rather than mass population. This grounding alongside social network

Manuscript received October 20, 2014; revised December 28, 2015; accepted December 31, 2015. Date of publication February 19, 2016; date of current version May 17, 2017. This work was supported in part by the National Natural Science Foundation of China under Grant 61170033 and in part by the National Key Technology Research and Development Program of China under Grant 2014BAD10B02.

S. Deng is with the College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China (e-mail: dengsg@zju.edu.cn).

L. Huang is with the Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100190, China (e-mail: huanglongtao@iie.ac.cn).

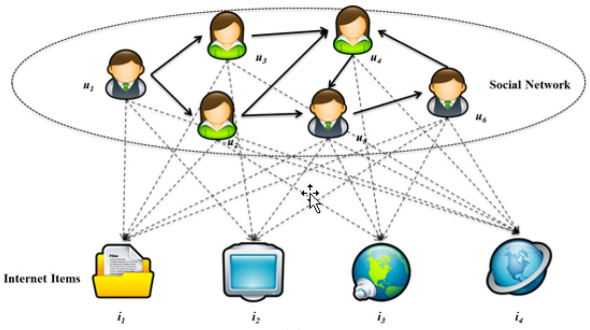
G. Xu is with the Advanced Analytics Institute, University of Technology at Sydney, Sydney, NSW 2007, Australia (e-mail: guandong.xu@uts.edu.au).

X. Wu is with the Department of Computer Science, The University of Vermont, Burlington, VT 05405 USA (e-mail: xwu@uvm.edu).

Z. Wu is with the College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China (e-mail: wzh@zju.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2016.2514368



(a)

	i_1	i_2	i_3	i_4
u_1	5	5		5
u_2	5		3	4
u_3	3	4		3
u_4			5	3
u_5	5	4	4	5
u_6	5	4	5	5

(b)

Fig. 1. Toy example of trust-aware social recommendation. (a) Real-world Internet item-rating scenario. (b) User-item-rating matrix.

analysis significantly motivates the algorithmic innovation, which can also tackle the aforementioned problems. Fig. 1 is an example to illustrate the key idea of trust-aware RS.

Example 1: Suppose a real-world Internet item-rating scenario, as shown in Fig. 1(a), in which users can rate items according to their personal experiences. In addition, the users' trust relationships can be reflected in the trust social network. The rating records of items by the users are shown in Fig. 1(b). In this example, user's missing preference scores, which are the unknown values in the rating matrix, such as $\langle u_4, i_1 \rangle$ and $\langle u_4, i_2 \rangle$, can be inferred by referring its neighbor's known ratings via similarity calculation given the whole rating data are reliable and sufficient enough. Suppose that the predicting missing values of $\langle u_4, i_1 \rangle$, and $\langle u_4, i_2 \rangle$ are 4.8 and 3.6, respectively, then item i_1 is better than i_2 for u_4. Apart from this traditional mechanism, the trusted-based RS can also treat the ratings of various trusted-users unequally. In other words, users' ratings are not only determined by their neighbor's rating, but also impacted by those of their trusted friends in a certain extent. Another important advantage of trusted-based RS is its capability of dealing with sparsity and cold-start. In this case, for a cold-start user, even we cannot determine the accurate neighbors of the user, we still can infer the user's preference via the friend network of the user given this information is available. Meanwhile, the advance of social network analysis provides credits to help improve recommendation. However, although the user trust relation is adequately utilized in the recommendation, the diversity of the user friendship has not been considered. Likewise, the preferences of friends in various domains contribute differently to form the recommendation for the users.

Recently, trust-aware RSs have attracted lots of attention [6]–[8], and the main stream in current approaches is to leverage the empowerment of machine learning exploits the matrix factorization (MF) technique to learn latent features for users and items from the observed ratings and trust relationships of users [9], [10]. Theoretical and empirical evaluations have verified its superiority compared with the other state-of-the-art approaches. In principle, MF can be considered as a prediction model learning process via estimating the model parameters from the observed training data. More concretely, it is indeed an optimization problem of determining the model parameters in order to best approximate the ground truth with the prediction. For an optimization problem, the initialization is sometimes a crucial issue to the quality of optimization. This inspires us to investigate this common difficulty in MF. Hence, this paper attempts to address these two questions simultaneously.

First, the current methods usually initialize the latent features of users and items based on quite simple mechanisms, such as random or zero initialization. Actually, from the perspective of optimization, MF-based methods are sensitive to the initialization of user and item latent feature matrices, since the minimization procedure in MF is nonconvex. A good initialization can lead to a better local minimum and improve the efficiency and accuracy of the learning process [34]. Hence, it is important to generate a good initialization for the latent features of users and items, which has been sidestepped or overlooked by most MF-based RS.

The determination of the initial user and item latent feature vectors should obviously be aligned with the learning algorithm applied. The learning of user latent and item matrix is a minimization of the loss function between the observed ratings R and the predicted ratings. Due to the high dimensionality of user and item, there may exist a set of local optimal values in the whole loss function space. How to select an appropriate initialization of user and item latent feature vector will undoubtedly influence the convergence of these matrices. If we can obtain the user and item latent feature in a low-dimensional space, it would be easier to deal with the initialization difficulty. To do this, we decide to utilize a deep learning method, i.e., deep autoencoder, which is an efficient approach to nonlinear dimensionality reduction. Deep autoencoder is quite suitable for the initialization phase, which can abstract the high-dimensional rating data to latent features through multiple layers of restricted Boltzmann machines (RBMs).

Second, the existing models in the current methods fail to consider the diversity of user trust networks, i.e., users trust different friends on different topics. Actually, people tend to form various communities according to their social relations in a social network, which is called community effect [31]. The community, no matter explicit or implicit, can embody the latent rules of the social network. Intuitively, the contribution of recommendation from different subsets of friends should be distinguished according to their interest similarities with the target user. That is, people in a community tend to trust each other and share common preferences with each other more than those in other communities. Hence, the community effect has great influence on the performance of recommendation methods in social networks.

Example 2: Given a user u_1 wants to see the movie *Transformers*, but he knows nothing about the movie. Then, he may consider his trusted friends' recommendations.

Among his trusted friends, u_2 and u_3 have rated this movie as 4 and 2, respectively, and u_1 trusts u_2 and u_3 at the same level. However, u_1 and u_2 both like science fiction action movies, while u_3 prefers literary types. Based on the information that *Transformers* is more relevant to scientific fiction movie than to literary movie, there is a very high probability that u_1 will think that the movie *Transformers* is a very good movie worth of watching. Thus, we think that u_1 and u_2 may form a community that favors on science fiction action movies and they are likely to share similar preferences on science fiction action movies. Ideally, if we can get all users' trust communities, we probably should distinguish the friends' recommendation from different trust communities.

The work in [36] tried to address the user community (i.e., so-called circle) in recommendations. It attempted to group users into different circles according to item categories. Then, it predicted ratings for one category by only using circles specific ratings to that category. Although this method can reflect community effect, the recommendations are heavily dependent on the attributes explicitly reflected from categories, which are sometimes unable to obtain in existing social networks. Hence, in this paper, we aim to integrate community detection with the trust-based recommendations. We first propose a community detection algorithm to identify user communities by mining their relationships in the social trust network, which cannot reflect users' explicit interests nor some implicit preferences. Then, we revamp the social trust ensemble-learning model by considering both the trusted friends' preference and the community effect.

In summary, we propose an MF-based approach for trust-aware recommendation in social networks, called Deep Learning based Matrix Factorization (DLMF). To overcome the first limitation described earlier, we examine the importance of the initialization of the MF methods. We propose a deep learning-based initialization method, in which deep autoencoder is utilized to pretrain the initial values of the parameters for our learning model. For the second limitation, we propose a social trust ensemble learning model, which not only considers trusted friends' recommendation, but also involves the community effect. Furthermore, we also provide a community detection algorithm to form community in a trust social network. To evaluate DLMF, we conduct a series of experiments based on the real-world data from Epinions and Flixster. The results show that DLMF can provide a better recommendation accuracy than the other state-of-the-art approaches.

The main contributions of this paper are as follows.

1) We develop the novel DLMF approach to tackle the trust-aware recommendation problem in social networks and propose a novel method to improve the quality of the initial vectors for MF by employing the deep autoencoder technique.
2) We revamp the MF model with social trust ensemble and community effect. In addition, a community detection algorithm based on trust relations in social networks is proposed.
3) We perform experiments on the real-life data set from Epinions and Flixster. Experimental results show that DLMF achieves significantly better recommendation accuracy in particular for sparse data and cold-start users in comparison with other methods.

The rest of this paper is organized as follows. Some related work is discussed in Section II. The problem definition is presented in Section III. We introduce our proposed approach in Section IV. We describe the experiments in Section V. Finally, we conclude this paper and present some directions for future work.

II. Related Work

Since this paper focuses on trust-aware recommendation, this section mainly reviews some significant work on trust-aware recommendation with different techniques. Integrating trust in recommendation has widely been explored in the memory-based approaches. The most common trust-aware RSs make users explicitly issue trust statements on other users. For example, Moleskiing used Friend-Of-A-Friend-files that contain trust information on a scale from 1 to 9 [11], and Epinions.com maintains reviews based on a network of trust by asking users to indicate which users they trust or distrust. Another well-known example is FilmTrust [12], an online social network that is combined with a movie rating system where users are encouraged to evaluate their friends' movie favors on a scale from 1 to 10. All these systems exploit the relationships in the trust network to determine which opinions or ratings should weigh more or less in the recommendation process. In the following, we first review some commonly used trust-aware recommendation system with memory-based approaches.

Golbeck [13] proposed an extended-breadth first-search method in the trust network for prediction called TidalTrust. TidalTrust finds all raters with the shortest path distance from the given user and then aggregates their ratings with the trust values between the given user and these raters as weights. To calculate the trust value between two users u and v, who are not directly connected, TidalTrust aggregates the trust value between u's direct neighbors and v weighted by the direct trust values of u and its direct neighbors. MoleTrust is similar to TidalTrust [14], but it only considers the raters with the limit of a given maximum depth. The maximum depth is independent of any specific user and item. The authors utilize a backward exploration to compute the trust value between u and v. It means that the trust value from u to v is the aggregation of trust values between u and users directly trusting v weighted by the direct trust values.

Jamali and Ester [7] proposed a random walk method named TrustWalker, which combined trust-based and item-based recommendations. The random walk model can efficiently avoid the impact of noisy data by considering enough ratings. TrustWalker considers not only ratings of the target item, but also those of similar items. The probability of using the rating of a similar item instead of a rating for the target item increases with increasing length of the walk. Their framework contains both the trust-based and item-based collaborative filtering recommendations as special cases. Their experiments show that their method outperforms other existing memory-based approaches.

The memory-based approaches suffer from the data sparsity issue in practice, because a typical user may only provide ratings for a limited number of items. Users should rate at least one common item in order to calculate their similarity. In this regard, only very limited information can be used, which will lead to poor quality of recommendation. In recent years, MF has widely been used in the model-based recommendation [15], [16]. However, these models do not consider the trust relationships among users. Recently, some model-based approaches have been proposed which use MF for trust-aware recommendation in social networks. In the following, we discuss model-based approaches.

Ma *et al.* [9] proposed an MF approach named Social Trust Ensemble (STE) for social network-based recommendation [9]. STE is based on the intuition that the decisions of each user on the items should include both the user's characteristics and the user's trusted friends' recommendations. The method is a linear combination of probabilistic MF approach and a social network-based approach. Experiments show that their model outperforms the basic MF-based approach and existing trust-based approaches. However, in their model, the feature vectors of direct neighbors of the given user u only affect the ratings of u but neglect the effect on the feature vector of u. Hence, this model does not handle trust propagation properly.

Compared with STE, Jamali and Ester [10] incorporated the mechanism of trust propagation into their recommendation model. They proposed a novel model-based approach named SocialMF for recommendation in social networks. Their model is an MF-based approach. Similar to the STE model, SocialMF learns the latent feature vectors of users and items. Different from STE, the feature vector of each user is dependent on the feature vectors of his direct neighbors in the social network. This allows SocialMF to handle the transitivity of trust and trust propagation, which is not captured by STE.

The proposed approach DLMF goes beyond the existing approaches by introducing the deep learning technique to provide trust-aware recommendation in social networks. Deep autoencoder is applied to learn the initial values of the MF model in the first learning phase, thus makes the second learning phase to provide a better quality of recommendation. Furthermore, the two-phase learning mechanism will more effectively address the cold-start issue by learning the latent features of users more precisely, which could not be appropriately addressed by most existing approaches.

Another key difference between the proposed DLMF approach and the existing social network recommendation systems lies in the recommendation models they provide. Most models in existing approaches simply combined the users' characteristics with their trust friends' recommendation based on different weights. However, our model explores the impact of the community effect, which has more complicated contribution to the quality of recommendation.

III. Problem Definition and Preliminary

As stated in [10], there is a set of users $U = \{u_1, u_2, \ldots, u_m\}$ and a set of items $I = \{i_1, i_2, \ldots, i_n\}$ in an RS. The ratings expressed by the users on items are given in a rating matrix $R = [R_{u,i}]_{m \times n}$. In this matrix, $R_{u,i}$ denotes the rating of user u on item i. $R_{u,i}$ can be any real number, but often ratings are integers in the range $[1, 5]$. In this paper, without loss of generality, we map the ratings $1, \ldots, 5$ to the interval $[0, 1]$ by normalizing the ratings. In a social rating network, each user u has a set S_u of direct neighbors, and $t_{u,v}$ denotes the value of social trust u has on v as a real number in $[0, 1]$. Zero means no trust, and one means full trust. Binary trust networks are the most common trust networks (Amazon, eBay, and so on). The trust values are given in a matrix $T = [T_{u,v}]_{m \times m}$. Nonzero elements $T_{u,v}$ in T denote the existence of a social relation from u to v. Note that T is asymmetric in general.

The social rating network can be presented as a graph where there are two types of nodes corresponding to users and items, respectively. The edges between the users correspond to the trust relations between the users, and the edges between the users and the items correspond to the ratings assigned. An example of a graph representation of social rating network is shown in Fig. 1.

Thus, the task of a trust-aware RS is given a user u and an item i for which $R_{u,i}$ is unknown, predict the rating for u on item i using R and T.

In the following, we first briefly review the basic MF approach for recommendation using only user-item-rating matrix. Then, in Section IV, we will introduce our proposed approach DLMF which employs deep learning and MF for recommendation in social rating networks.

Matrix factorization model is an efficient mechanism for predicting missing values and becomes more and more popular in RSs [27]. This model maps both the users and the items to a joint latent feature space of a low dimension k, so that user-item interactions can be modeled as inner products in that space. The premise behind the MF technique is that there are only a few key features affecting the user-item interactions, and a user's interactive experience is impacted by how each feature applies to the user [16]. In general, each user u corresponds to a column vector $P_u \in \mathbf{R}^k$, and each item i corresponds to a column vector $Q_i \in \mathbf{R}^k$. For a given user u, the elements of P_u measure the extent of interest the user has in items that are high on the corresponding features. For a given item i, the elements of Q_i measure the extent to which the item possesses those features. Then, the m users and n items form the user latent feature matrix $P \in \mathbf{R}^{k \times m}$ and item latent feature matrix $Q \in \mathbf{R}^{k \times n}$, respectively. The resulting dot product, $P_u^T Q_i$, models the interaction between user u and item i. Hence, the user-item-rating matrix R can approximately be divided into two parts P and Q with k-dimensional features constraints

$$R \approx P^T Q. \tag{1}$$

The goal of MF is to learn these latent variables by minimizing the following term:

$$L(R, P, Q) = \frac{1}{2} \min_{P,Q} \sum_{u=1}^{m} \sum_{i=1}^{n} I_{ui} \left(R_{u,i} - P_u^T Q_i\right) + \frac{\lambda_1}{2} ||P||_F^2 + \frac{\lambda_2}{2} ||Q||_F^2 \tag{2}$$

where I_{ui} is the indicator function that equals 1 if user u rated item i and equals 0 otherwise. λ_1 and λ_2 are regularization terms to avoid model overfitting. $||\cdot||_F^2$ denotes the Frobenius norm. The initial values of P and Q are always generated randomly or manually. Then, in each iteration, P and Q are updated by employing the stochastic gradient descent technique as follows:

$$P_u' = P_u - \gamma_1 \frac{\partial L}{\partial P_u}$$
$$Q_i' = Q_i - \gamma_2 \frac{\partial L}{\partial Q_i} \tag{3}$$

where $\gamma_1 > 0$ and $\gamma_2 > 0$ are set as the learning rates. To reduce the model complexity, we set $\gamma_1 = \gamma_2$ in our experiments. Mnih and Salakhutdinov [17] provided a probabilistic foundation for regularizing the learned variables, which has been employed by some recent approaches [9], [10].

IV. Trust-Aware Recommendation Approach

This section presents our approach DLMF in detail to incorporate deep learning into an MF model for trust-aware recommendation in social networks. In our proposed approach DLMF, we utilize deep autoencoder to learn the initial values of latent features of users and items at first, then using the learned results for the learning of minimizing the objective function. For our objective function in the MF model, we consider both the users' characteristics and their trusted friends' recommendations. Besides, a trust-aware regularization term is added to describe the trust propagation. In the following parts, more details of DLMF are introduced.

A. Pretraining With Deep Autoencoder

Due to the nonconvex optimization formula of MF, there is no guarantee that both factorized matrices (P and Q) are optimally determined [38]. Furthermore, MF can converge to different local optimum corresponding to different initial values of P and Q. Therefore, if the initialization values are set properly, the results would be nearly optimum compared with the situation that the initial values are determined far from the global optimum. In this section, we analyze how to employ deep autoencoder to pretrain the rating matrix and learn the initial values of the latent features of users and items.

In the rating matrix $R = [R_{u,i}]_{m\times n}$, each row corresponds to the rating experience on all items from one user. Thus, we can denote user vectors by $U_{n\times m} = [U_1, \ldots, U_i, \ldots, U_m] = R^T$, in which each vector is denoted by $U_i = [R_{i,1}, \ldots, R_{i,j}, \ldots, R_{i,n}]^T$. Similarly, item vectors can be denoted by $I_{m\times n} = [I_1, \ldots, I_i, \ldots, I_n]$, where each vector is $I_i = [R_{1,i}, \ldots, R_{j,i}, \ldots, R_{m,i}]^T$. In the real world, the number of users and items is both extremely large. Hence, the MF method maps user vectors and item vectors to a joint latent factor space of a low dimensionally k, i.e., $P_{k\times m}$ and $Q_{k\times n}$. In the most current MF methods, $P_{k\times m}$ and $Q_{k\times n}$ are initialized randomly. After multiple iterations, these methods converge to a local optimum that depends on the starting point. In this paper, we follow the hypothesis that an appropriate initialization of latent features yields that the MF methods produce more precise feature vectors and, therefore, provide more accurate predictions [35].

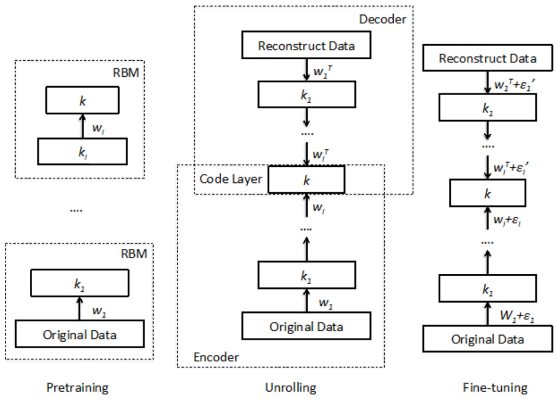


Fig. 2. Framework of deep autoencoder.

When investigating the feature vectors of accurate MF models, we can observe that similar users (e.g., users rating similar scores for the same items) have similar user feature vectors. Based on this observation, we can intuitively conclude that if the initialization of feature vectors characterized the similarities of users and items, more precisely, it will result in more appropriate models. Thus, we try to get the initial values of $P_{k\times m}$ and $Q_{k\times n}$ by extracting features from the original user vectors and item vectors $U_{n\times m}$ and $I_{m\times n}$. It is worth to note that the original user vectors and item vectors are typically very sparse and the initial vectors should be compliant with the feature size of the MF model [37]. Hence, the goal of pretraining turns out to dimensionally reduction or feature extraction.

The autoencoder is an artificial neural network used for learning efficient coding, which aims at learning a compressed representation (encoding) for a set of high-dimensional data [28]. Hinton and Salakhutdinov [29] have developed a pretraining technique for training many-layered deep autoencoders and then using a back-propagation technique to fine-tune [29]. The whole system of deep autoencoder is shown in Fig. 2, which consists of an encoder network and a decoder network. The encoder network targets to transform the original data with high dimensionality into a low-dimensional code. In addition, the decoder network, which can be regarded as the inverse process of the encoder network, is to reconstruct the original data from the code. The joint part of the two networks is called code layer, which is the core of the whole system and determines the intrinsic dimensions of the original data.

The main working process of deep autoencoder is started with learning weights for the two networks using RBM [30] and then used backpropagation of error derivatives to train the two networks together by minimizing the deviation between the original data and its reconstruction. However, RBM is only applicable for binary data but not ideal for modeling continuous data like the rating matrix in this paper. Therefore, we utilize continuous RBM (CRBM) for pretraining weights.

CRBM consists of a visible layer and a hidden layer, which correspond to the input data and the output data, respectively.

Take user vectors as an example, $U_{n\times m}$ is the input and $U'_{p\times m}$ is the output. Thus, each n-dimensional user vector is encoded as p-dimensional vectors. In addition, there are n visible units and p hidden units. For a visible unit v_i and a hidden unit h_j, w_{ij} denotes the weight between them and $w_{ij} = w_{ji}$. Then, the main steps of encoding stage with CRBM are in the following.

1) For each data vector v from $U_{n\times m}$, h for the hidden units can be achieved by

$$h_j = \varphi_j\left(\sum_i w_{ij}v_i + \sigma N_j(0,1)\right)$$
$$n_j = \sigma N_j(0,1) \tag{4}$$

where the set v_i denotes the set of visible units which contribute to the input of h_j. $N_j(0,1)$ represents a Gaussian random variable. The constants σ and $N_j(0,1)$ form a noise input component $n_j = \sigma N_j(0,1)$ according to a probability distribution

$$p(n_j) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left(\frac{-n_j^2}{2\sigma^2}\right). \tag{5}$$

In (4), $\varphi_j(x)$ is a sigmoid function with asymptotes at θ_L and θ_H

$$\varphi_j(x) = \theta_L + (\theta_H - \theta_L)\frac{1}{1+\exp(-a_j x)} \tag{6}$$

where a_j controls the slope of the sigmoid function.

2) Reconstruct each visible vector as v with the following equation that is similar to (4):

$$v'_i = \varphi_i\left(\sum_j w_{ji}h_j + \sigma N_i(0,1)\right) \tag{7}$$

where the set h_j denotes the hidden set of units which get the output of v_i.

3) Recompute the hidden states as h' using (4) with v' instead of v.

4) Update the weights w_{ij} and the noise controllers a_j using minimizing contrastive divergence

$$\Delta w_{ij} = \eta_w\left(v_i h_j - v'_i h'_j\right) \tag{8}$$
$$\Delta a_j = \frac{\eta_a}{a_j^2}\left(h_j^2 - h_j'^2\right). \tag{9}$$

Actually, there could be multiple CRBMs to encode the n-dimensional user vectors to the final k-dimensional vectors. Each input data of a CRBM is the output of the previous CRBM, and the top CRBM reaches the code layer. The fine-tune stage is relatively simple, and the backpropagation is utilized to minimize the root mean squared reconstruction error $(\sum_i (v_i - \hat{v}_i^2)^2)^{(1/2)}$ until an optimal reconstruction is reached. Finally, the optimal autoencoder is trained, and the k-dimensional vectors of users and items are achieved.

B. Social Trust Ensemble

In this section, we investigate how the social trust networks affect users' decisions on selecting items, and extend the basic MF model by involving the recommendations of trusted friends.

As discussed in Section IV-A, the trust values are given in a matrix $T = [T_{u,v}]_{m\times m}$. Nonzero elements $T_{u,v}$ in T denote the existence of a social relation from u to v. In addition, we regard the value of $T_{u,v}$ as the trust degree of the user u on the user v, note that $T_{u,v}$ is asymmetric in general. In most existing work, the trust degree is usually assigned by users explicitly. However, in most real-world online social networks (such as Facebook, Epinions, and Flixster), there is no explicit value that measures the extent of users' trust relations. Hence, the first we do is to propose a new model of trust degree no matter the trust values are assigned explicitly or not.

Section I has mentioned that people always prefer to rely on their friends' recommendations, because the comments from friends are more trustworthy. However, the recommendation from trusted friends is not absolutely suitable for the target user, since they may have different tastes, preferences, or habits. Then, we associate the preference similarity to model trust degree $T_{u,v}$ as follows:

$$T_{u,v} = \frac{\text{sim}(u,v)\times\text{trust}(u,v)}{\sum_{s\in S_u}\text{sim}(u,s)\times\text{trust}(u,s)} \tag{10}$$

where $\text{sim}(u,v)$ is the similarity of the two users by calculating the cosine similarity of their corresponding vectors P_u and P_v. $\text{trust}(u,v)$ is the assigned trust value by user u. If the value is not assigned explicitly, it equals 1 if the user u links to the user v and 0 otherwise. S_u is the set of users whom u directly links to. Note that this equation is only applicable for two users which are linked directly. For the users which are not linked directly, we use multiplication as the trust propagation operator to calculate the trust degree. In addition, only the shortest path is considered if multiple trust propagation paths exist.

After calculating the trust degree of users, we will illustrate how to ensemble the trust degree to the MF model. In the basic MF approach, the estimated rating of the item i from the user u is only interpreted by the user's favor on the item as $\hat{R}_{u,i} = P_u^T Q_i$. As the analysis above, recommendations on item i from user u's trusted friends S_u should also be considered. Hence, we extend the form of estimated ratings by considering both the above factors as follows:

$$\hat{R}_{u,i} = P_u^T Q_i + \sum_{v\in S_u} T_{u,v}\left(P_v^T Q_i + \Delta_{u,v} - P_u^T Q_i\right). \tag{11}$$

In this equation, rather than involving the favors of u's trusted friends directly, we consider the differences of the favors of u's trusted friends and u's favors. This means, u's favor on i $(P_u^T Q_i)$ is a base term, and is adjusted by the deviations from his trusted friends' favors. $\Delta_{u,v}$ means the average bias on ratings between u and v. For example, user u is a generous rater, who always provides high ratings. While v is a critical rater, who always gives low ratings and their average bias on ratings are around 2. Then, for an item i, the ratings from them are 5 and 3, respectively. It is intuitive that they have different opinions on the item i from the ratings directly. However, if the bias on ratings is considered, 3 is almost the highest ratings given by user v, then we may conclude that both u and v think the item i is good.

Furthermore, there also exist biases on users and items, respectively [16]. For example, suppose that we want to predict user u's rating on the item i. The average rating over all movies avg is 3. Besides, item i is better than an average item and tends to be rated 0.5 higher than the average. On the other hand, user u is a critical rater, who tends to rate 0.3 lower than the average. Thus, the estimate for item i from user u would be 3.2 ($3 + 0.5 - 0.3$). Thus, (5) is extended with biases of users and items as follows:

$$\hat{R}_{u,i} = P_u^T Q_i + \sum_{v \in S(u)} T_{u,v}\left(P_v^T Q_i + \Delta_{u,v} - P_u^T Q_i\right) + \text{avg} + \text{bias}_u + \text{bias}_i \quad (12)$$

where the parameters bias_u and bias_i indicate the deviations of user u and item i, respectively.

C. Regularization With Community Effect

As mentioned earlier, social networks demonstrate a strong community effect. Users in social networks have a tendency to form groups of closely knit connections. The groups are also called communities, clusters, or cliques in different contexts. Furthermore, people in a group tend to trust each other and share common preferences with each other more than those in other groups. Hence, in this section, we discuss how to incorporate the community effect in a trust social network as regularization terms to revamp the proposed MF model in detail.

Graph mining techniques have widely been used for community detection in social networks, as they are effective in identifying groups that are hidden in the underlying data. One important type of community detection is to identify cliques based on the reachability between actors in a network. Inspired by the existing community structure *n-clique* in social science, we propose the *n-trust-clique* to identify the cliques in a social network based on the trust relationship between the users and develop an algorithm TrustCliques to detect cliques. The reason that we choose *n-clique* is that this kind of methods helps to detect overlapping communities. Besides, it is worth to note that the reason that we do not directly apply *n-clique* on the social network is that the link information itself will lead to low detection accuracy, while the trust information helps guarantee good performance. In what follows, we first introduce an important parameter, collectivity, used by the proposed TrustCliques algorithm in Section IV-C1. Then, we give the details of the TrustCliques algorithm in Section IV-C2. At last, we present the regularization terms based on community effect in Section IV-C3.

1) Key Definitions for Community Detection: The notion of *n-clique* is a typical concept for cliques in social science [32], which is defined as follows.

Definition 1 (n-Clique): Given a network G, an *n-clique* is a maximal subgraph in which the largest distance of each pair of nodes is no greater than n. That is

$$\text{dist}(v_i, v_j) \leq n \quad \forall v_i, v_j \in G.$$

Note that the distance is defined in the original network. Thus, the geodesic is not necessarily included in the group structure. Therefore, an *n-clique* may have a diameter greater than n or even become disconnected.

In order to extend the notion of *n-clique* to handle the trust relationships in the social network, we propose a parameter collectivity. The definition is as follows.

Definition 2 (Collectivity): Given a user trust matrix T and the corresponding trust network G_T, assume that the users have been clustered into a set of cliques C_i. The collectivity of a clique C_i is defined as

$$\text{Col}_i = \frac{\sum_{u,v \in C_i} T_{u,v}}{\sum_{u \in C_i, v \notin C_i} T_{u,v}}.$$

Thus, the global collectivity of user collection, the trust network, is defined as follows:

$$\text{Col} = \log\left(\frac{\sum_{C_i \in \text{Clqs}} \text{Col}_i}{|\text{Clqs}|}\right)$$

where Clqs is the set of cliques, and the log operator is employed to ensure that global collectivity decreases linearly with the number of cliques.

Based on the parameter collectivity, we propose a new community structure called *n-trust-clique*, which is defined as follows.

Definition 3 (n-Trust-Clique): Given a trust network G_T, an *n-trust-clique* is a subgraph in which the largest distance of each pair of nodes is no greater than n. At the same time, the generation of cliques in G_T should guarantee the maximum value of the global collectivity Col.

To achieve the maximum value of the global collectivity, Col is an Non-deterministic Polynomial-hard problem. Therefore, we utilize optimization methods to get approximate *n-trust-cliques* in the following. The larger value of Col means that users in the same clique have higher trust degrees with each other than with users in other cliques. That is, users in the same cliques are more likely to share common preferences and there is greater difference between the users in different cliques. Hence, if the clique preference feature is more obvious, the recommendation prediction is more precise.

2) Algorithm for Community Detection: Based on the previous definitions, we present the details of the proposed algorithm TrustCliques in Algorithm 1. This algorithm is inspired from [33], which combines the greedy techniques and the simulated annealing techniques. In particular, the algorithm runs l iterations, and each iteration produces a partition result Clqs, which is initialized as the user set U and each user forms a clique. For each C_i in Cliqs, J is the set of C_i's connected vertices. We assume to merge the clique C_i and $v_j \in J$, calculate the ΔCol for the merge and pick the merged clique C_t with the maximum ΔCol (lines 4–14 in Algorithm 1). If $\Delta\text{Col}_{\max} > 0$, we accept this change and update Clqs. Otherwise, we give a probability *prob* to accept this (lines 16–20). This is based on the stimulated annealing technique. For simplification, we just set the probability to a real-valued number between 0 and 1, e.g., 0.3. After i runs over all cliques in each iteration, we get a new partition Cliqs. Then, we store the partition information and the corresponding collectivity. After l iterations, we get l different

Algorithm 1 TrustCliques Algorithm

Require:
clique size n, iteration limitation l,
user set U, user trust matrix T;
Ensure:
cliques collection *Clqs*;
1: $Clqs = U$, $n_c = |Clqs|$;
2: **while** $n_c > 1$ or the number of iterations $< l$ **do**
3: **for** $i = 1$ to n_c **do**
4: C_i = the ith clique in *Clqs*,
J = the set of C_i's connected vertices;
5: $\Delta Col_{max} = 0$;
6: **for** each $v_j \epsilon J$ **do**
7: $C_i' = C_i \cup v_j$;
8: **if** C_i' satisfies the first condition of *n-trust-clique* **then**
9: calculate ΔCol for the change;
10: **if** $\Delta Col > \Delta Col_{max}$ **then**
11: $\Delta Col_{max} = \Delta Col$;
12:: $C_t = C_i'$;
13: **end if**
14: **end if**
15: **end if**
16: **if** $\Delta Col_{max} > 0$ or $Random(0, 1) > prob$ **then**
17: delete clique C_i from *Clqs*;
18: add C_t to *Clqs*;
19: store current partition result *Clqs* and the corresponding *Col*;
20: **end if**
21: **end if**
22: $n_c = |clqs|$, iteration number++;
23: **end while**
24: **return** optimal *Clqs* with the maximum value of *Col*;

partition results. Finally, we choose the partition result with the maximum value of collectivity as the final result (line 24 in Algorithm 1).

3) Regularization With Community Effect: As mentioned earlier, it is intuitive that users tend to share similar preferences on items with their trusted friends in the same clique. We regard these trusted friends as the neighbors of the given user. These neighbors can contribute more meaningful information to improving prediction accuracy. For a given user u, a set of neighbors $N(u)$ can be defined as follows:

$$N(u) = \{v | v \in C \wedge u \in C, u \neq v\} \tag{13}$$

where C is a clique that the given user u belongs to. Note that u may belong to not only one clique, we take all the cliques u belongs to into consideration.

With the help of neighborhood information, we propose a regularization term for ratings prediction. Due to community effect, the behavior of a user u would be affected by his neighbors $N(u)$. This indicates that the difference of user feature vectors in the neighborhood should be minor. Then, this idea can be expressed mathematically by minimizing the following form:

$$\left|\left| P_u - \frac{1}{|N(u)|} \sum_{v \in N(u)} T_{u,v} P_v \right|\right|_F^2. \tag{14}$$

The above regularization term is used to minimize the preferences between a user u and his neighborhood to an average level. That means, if the neighborhood of user u is $N(u)$, then we can assume that u's preferences (feature vector P_u) should be similar to the general preferences of all neighbors in $N(u)$. Then, we add this regularization term in our proposed objective function to revamp the MF model as follows:

$$\begin{aligned} L(R, T, P, Q) = \frac{1}{2} \min_{P,Q} \sum_{u=1}^{m} \sum_{i=1}^{n} I_{ui} (R_{u,i} - \hat{R}_{u,i}) \\ + \frac{\mu}{2} \sum_{u=1}^{m} \left|\left| P_u - \frac{1}{|N(u)|} \sum_{v \in N(u)} T_{u,v} P_v \right|\right|_F^2 \\ + \frac{\lambda_1}{2} ||P||_F^2 + \frac{\lambda_2}{2} ||Q||_F^2 \end{aligned} \tag{15}$$

where the nonnegative parameter μ is used to control the importance of the regularization term. To reduce the model complexity, we set $\lambda_1 = \lambda_2$ in our experiments. We can observe that this objective function considers all the users, and thus, it is aiming at minimizing the global difference within different neighborhoods. Similar to the basic MF model, the global minimum of L cannot be achieved due to the nature of its inner structure [38]. We can find a local minimum of the objective function utilizing gradient descent on P_u and Q_i for all users u and all items i

$$\begin{aligned} \frac{\partial L}{\partial P_u} = \sum_{i=1}^{n} I_{ui} (\hat{R}_{u,i} - R_{u,i}) \left(Q_i - \sum_{v \in S(u)} T_{u,v} Q_i \right) \\ + \mu \left(P_u - \frac{1}{|N(u)|} \sum_{v \in N(u)} T_{u,v} P_v \right) + \lambda_1 P_u \end{aligned} \tag{16}$$

$$\begin{aligned} \frac{\partial L}{\partial Q_i} = \sum_{u=1}^{m} I_{ui} (\hat{R}_{u,i} - R_{u,i}) \left(P_u^T + \sum_{v \in S(u)} T_{u,v} \left(P_v^T - P_u^T \right) \right) \\ + \lambda_2 Q_i. \end{aligned} \tag{17}$$

V. Experiments

In this section, we conduct several experiments to compare the recommendation qualities of our DLMF approach with the other state-of-the-art trust-aware recommendation methods. Our experiments aim at addressing the following questions.

1) How do the model parameters affect the prediction accuracy?
2) How does the pretraining phase affects the prediction accuracy?
3) How does the community effect affects the perdition accuracy?
4) How does DLMF compared with the existing state-of-the-art trust-aware recommendation algorithms with different sparsities of data sets?
5) Can DLMF performs well with cold-start users?

TABLE I
STATISTICS OF DATA SETS

	Epinions	Flixster
Users	49,290	1,049,445
Items	139,738	492,359
Ratings	664,824	8,238,597
Trust Relations	487,181	26, 771,123

A. Experimental Setup

1) Data Set: In the trust-aware recommender research domain, there is not much publicly available and suitable test data [18]. In our experiments, we mainly use the following two data sets: 1) the Epinions data set and 2) the Flixster data set.

Epinions is a well-known product review Web site established in 1999 [19]. Users can rate products from 1 to 5 and submit their personal reviews. These ratings and reviews will influence other customers when they make a decision on whether to buy a product. Besides, users are also allowed to specify whom to trust and build a social trust network. In social trust network, reviews and ratings from a user can be consistently found to be valuable by his trustees. Flixster is a social networking service in which users can rate movies. Users can also add some users to their friend list and create a social network. Thus, Epinions and Flixster are both ideal source for our experiments. We use the data set of Epinions [20] and the data set of Flixster [10].

Table I shows the statistics of the two data sources. It shows that the user-item-rating matrix of Epinions is much sparser than Flixster. Thus, the Epinions data set can evaluate the performance of our approach with high sparsity.

2) Evaluation Metrics: Our experiments are developed by Java, and the execution environment is Intel Core2 P7370 2.0 GHZ with 4GB RAM, Windows 7, and jdk1.6.0.

We adopt the root mean squared error (RMSE) to measure the error in recommendation, which has widely been used in recommendation research [9], [10]

$$\text{RMSE} = \sqrt{\frac{\sum_{u,i} (R_{u,i} - \hat{R}_{u,i})^2}{N}} \tag{18}$$

where $R_{i,j}$ is the actual rating the user u gave to the item i and $\hat{R}_{u,i}$ is the predicted rating the user u gave to the item i. N denotes the number of tested ratings. The smaller the value of RMSE is, the more precisely the recommendation is.

Meanwhile, some recommendation mechanisms may not be able to predict all the ratings in test data for the high sparsity of the data set. Involving trust can enhance the coverage without sacrificing the precision. Therefore, we use the metric coverage to measure the percentage of pairs of ⟨user, item⟩ that a predicted value can be generated. For a pair of ⟨user, item⟩, if the recommender cannot find a prediction on the rating, then it means that the recommender can not cover this pair of ⟨user, item⟩

$$\text{coverage} = \frac{S}{N} \tag{19}$$

where S denotes the number of predicted ratings, and N denotes the number of tested ratings.

To combine RMSE and coverage into a single evaluation metric, we compute the F-measure. Therefore, we have to convert RMSE into a precision metric in the range [0, 1]. The precision is denoted as follows:

$$\text{precision} = 1 - \frac{\text{RMSE}}{4}. \tag{20}$$

In this equation, 4 is the maximum possible error, since the values of ratings are in the range [1, 5]

$$F - \text{Measure} = \frac{2 \times \text{precision} \times \text{coverage}}{\text{precision} + \text{coverage}}. \tag{21}$$

B. Impact of Model Parameter μ

In our DLMF approach, the parameter μ controls how much the trust-aware regularization terms influence to the objective function of (12). A larger value of μ indicates that the social trust network has more impact on the characteristics of users. An extremely small value of μ makes our model focus on the basic MF model and weaken the impact of the trust-aware regularization terms. However, an extremely large value of μ leads to such a model that the social trust information dominates the prediction, which would potentially limit the prediction accuracy. In this section, we study how the changes of μ can impact the prediction accuracy. We change μ from 0 to 50. For other parameters, we utilize the grid-search method to find the best combination values. We set dimensionality $k = 80$, $\lambda_1 = \lambda_2 = 0.1$, and $\gamma_1 = \gamma_2 = 0.01$. For the community consideration, we use *2-trust-cliques*.

Fig. 3 compares the RMSE of our model with different values of μ. From the results, we can see that as μ increases, the RMSE value decreases at first. When μ passes over a certain threshold, the RMSE value increases again. We observe that the threshold is around $\mu = 10$ for Epinions data set and $\mu = 5$ for Flixster data set. Thus, we can conclude that DLMF has its best performance on Epinions with $\mu = 10$ and on Flixster with $\mu = 5$. The results also indicate that solely using the MF and trust relationship cannot lead to a better prediction accuracy unless an appropriate combination.

C. Analysis of Pretraining Phase

1) Impact of Pretraining: In our method, we propose to initialize the latent features of users and items P and Q utilizing deep autoencoder. While most of existing MF methods for recommendation use simple initialization, i.e., P and Q are initialized as dense matrices of random numbers. To study the advantages of initialization with deep autoencoder, we compare our initialization method with other four classic methods—random initialization, zero initialization, K-means initialization, and normalized-cut (Ncut) initialization [39]. For the learning model, we set $\mu = 10$ for Epinions data set and $\mu = 5$ for Flixster data set, $k = 80$. For the community consideration, we use *2-trust-cliques*. Fig. 4 shows that our proposed initialization method outperforms the other two. This result is caused by two reasons: 1) our method starts with the original feature vectors to extract features, which are much

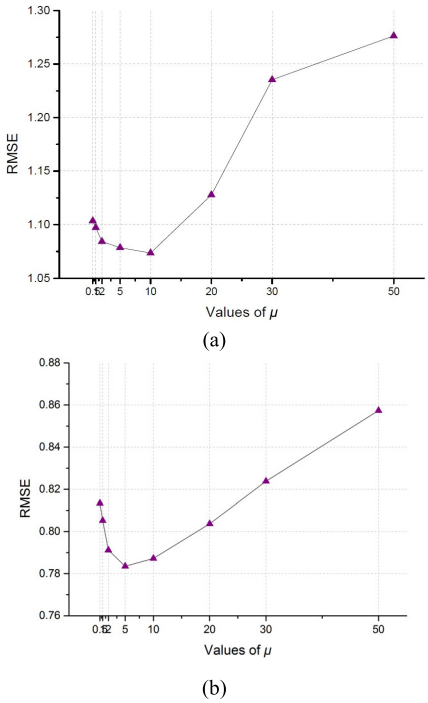


Fig. 3. Impact of different values of μ on the prediction accuracy. RMSE decreases at first and increases when μ passes over a certain threshold. (a) Epinons data set. (b) Flixster data set.

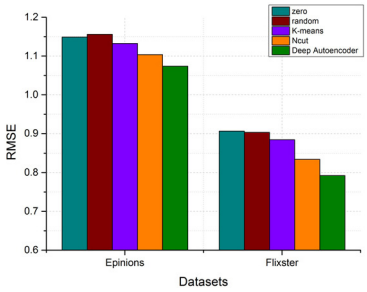


Fig. 4. Impact of initialization for learning models. Deep autoencoder can provide better initializations than methods compared.

closer to the global minimum of the learning model and 2) the initial vectors achieved by our method can reflect the original relations of the similarities of users and items better than the other two methods.

Furthermore, we also evaluate the total time cost to learn the prediction model with different initialization methods. From the comparison result in Table II, we can observe that our method costs around 20% more time than the others in average. The increased time cost is caused by two reasons: 1) it would take more time to learn the initial values by deep autoencoder than just generating them zero or randomly and 2) the iteration number to converge is increased, which can avoid converging to a bad local optimum too early. However, the time is mainly used to train the prediction model, which can be executed offline. Therefore, it will not impact users' interaction experience. However, the improvement of prediction accuracy can bring more benefit.

TABLE II
TOTAL TIME TO LEARN THE PARAMETERS OF MODELS

Initialization Method	Epinions	Flixster
zero	22min	53min
random	21min	53min
K-means	24min	57min
Ncut	25min	60min
deep autoencoder	27min	72min

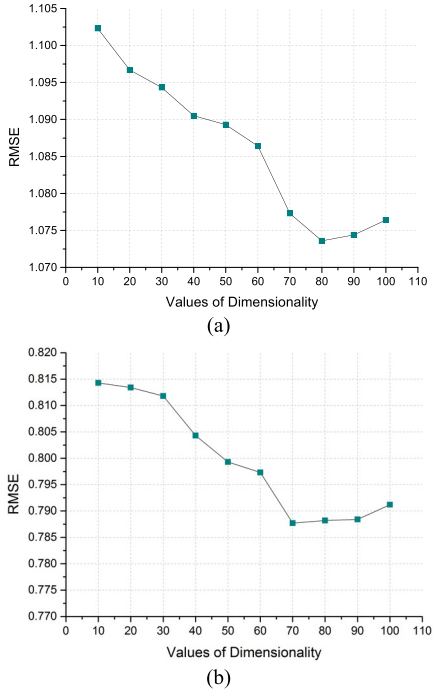


Fig. 5. Impact of different values of dimensionality k on the prediction accuracy. RMSE decreases at first and increases when k passes over a certain threshold. (a) Epinons data set. (b) Flixster data set.

2) Impact of Dimensionality k: The purpose of the pretraining phase of our method is to reduce the dimensionality of the user-item matrix and extract features to model the characteristics of users and items. Then, dimensionality controls the number of factors for MF. To study the impact of dimensionality, we utilize the grid-search method to find the best combination values. We set $\mu = 10$ for Epinions data set and $\mu = 5$ for Flixster data set, $\lambda_1 = \lambda_2 = 0.1$ and $\gamma_1 = \gamma_2 = 0.01$. For the community consideration, we use *2-trust-cliques*. Fig. 5 shows that with the increase of dimensionality, the value of RMSE dramatically decreases at first. However, the value of RMSE increases when dimensionality goes above a certain threshold (around 80 for Epinions and 70 for Flixster). Two reasons cause this observation: 1) the improvement of prediction accuracy confirms the intuition that a relative larger dimension leads to better results and 2) when the dimensionality surpasses a certain threshold, it may cause the issue of overfitting, which turns out to degrade the prediction accuracy.

3) Impact of Pretraining Algorithms: In our proposed method, we utilize deep autoencoder to reduce dimensionality and extract features of users and items from the rating matrix.

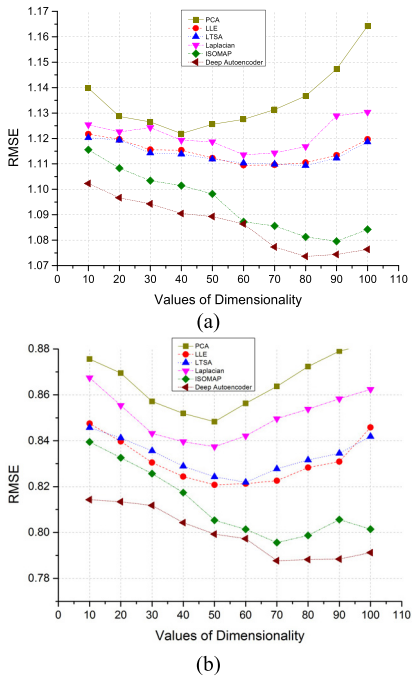


Fig. 6. Comparison results of different feature extraction algorithms. (a) Epinons data set. (b) Flixster data set.

Actually, there are many classic algorithms for feature extraction, such as principal component analysis (PCA) [21], locally linear embedding [22], local tangent space alignment [23], Laplacian eigenmap [24], and ISOMAP [25]. Therefore, in this section, we compare the performance of applying these algorithms with deep autoencoder for pretraining.

Except for PCA, the other algorithms need to specify the parameter K (the number of the nearest neighbors). We set $K = 8$ for the experiments, which seems to perform well for the experiments.

For the comparison, we utilize the grid-search method to find the best combination values. We set $\mu = 10$ for Epinions data set and $\mu = 5$ for Flixster data set, $\lambda_1 = \lambda_2 = 0.1$ and $\gamma_1 = \gamma_2 = 0.01$. For the community consideration, we use *2-trust-cliques*. In addition we also tune the dimensionality k from 10 to 100. From the comparison results shown in Fig. 6, we can observe that involving pretraining phase can improve the prediction accuracy very much compared with the criterion algorithm. In addition, deep autoencoder can provide lower RMSE compared with the other feature extraction algorithms. That means deep autoencoder is more suitable for trust-aware recommendation.

D. Impact of Community Effect

For our method, we involve a trust-aware regularization term to incorporate the community effect for recommendation. In order to evaluate the impact of the community effect in the trust social network, we compare the recommendation behavior between the model with and without the proposed regularization term. Besides, we also take the model using *n-clique* to generate cliques for comparison. For our method, we set $\mu = 10$ for Epinions data set, $\mu = 5$ for Flixster data set, $k = 80$ for Epinions data set, and $k = 70$ for

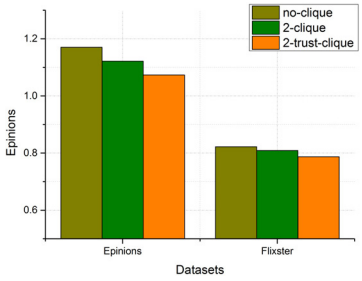


Fig. 7. Impact of community effect.

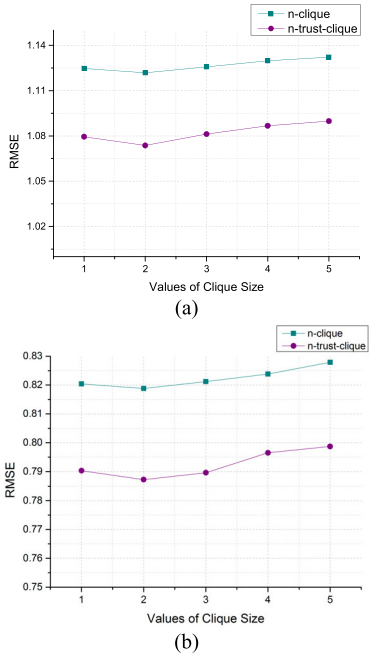


Fig. 8. Impact of different values of clique size n on the prediction accuracy. (a) Epinons data set. (b) Flixster data set.

Flixster data set. In addition, we generate cliques with *2-clique* and *2-trust-clique*, respectively. Fig. 7 shows the comparison results and we can observe that the model with the proposed regularization term can provide better recommendation precision. Furthermore, the model using *2-trust-clique* performs better than *2-clique*, which means the generating cliques with the proposed TrustCliques algorithm is more suitable for the trust-aware recommendation problem.

The value of clique size n decides the size of neighbors. If the value of n is quit small, only those neighbors which are quite trustworthy can be identified. If the value of n is quite large, the neighbors would be incorporated to a large extent. We set $\mu = 10$ for Epinions data set, $\mu = 5$ for Flixster data set, $k = 80$ for Epinions data set, and $k = 70$ for Flixster data set. In addition, we tune the clique size n from 1 to 5. Fig. 8 shows the impact of clique size n on the prediction accuracy. We can find that when n increases, the RMSE value decreases at first. However, when n passes over a threshold, the RMSE value increases again. This observation can be explained as when n is smaller than a certain threshold, there are few neighbors contributing to missing ratings predictions, which prevents user to fully absorb the wisdom of crowds.

When n is larger than a certain threshold, the neighbors contain much noise, even though the sample size is large enough. The above two cases will both lead to lower prediction accuracy.

E. Comparison Results

In order to evaluate the performance improvement of DLMF, we compare our method with the following state-of-the-art methods.

1) *UserCF:* This method is the classic user-based collaborative filtering method, which makes prediction according to users' similarity.
2) *ItemCF:* This method is the classic item-based collaborative filtering method, which captures similar item characteristics to make prediction.
3) *TidalTrust:* This method is proposed in [20], which generates predictive ratings based on a trust inference algorithm.
4) *MoleTrust:* This method is proposed in [14], which predicts the trust score of source user on target user by walking the social network starting from the source user and by propagating trust along trust edges.
5) *BMF:* This method is the basic MF approach proposed in [16], which does not consider the social network.
6) *STE:* This method is proposed in [9], which employs the social trust network to make recommendations for users. The parameter α for STE is set as 0.4 in our comparison experiments, which is the optimum value to handle the data sets.
7) *SocialMF:* This method is proposed in [10], which incorporates the mechanism of trust propagation into the recommendation model. For our comparison experiments, we set the parameter $\lambda = 5$ for SocialMF, which can provide the best results in this experiment.
8) *GWNMF:* This method is proposed in [40], which is a collaborative filtering method based on graph regularized weighted nonnegative MF.

In order to get a fair comparison, we use the grid-search method to get the best performance of these compared algorithms when processing the data sets in this paper. Thus, all the compared algorithms are adjusted to get the best performance with the data sets.

1) Comparison With All Users: In this section, we compare the performance of the above methods with ours using different training data, thus the behavior of these methods with different data sparsity is shown. For our method, we set $\mu = 10$ for Epinions data set, $\mu = 5$ for Flixster data set, $k = 80$ for Epinions data set, and $k = 70$ for Flixster data set. From the comparison results shown in Table III, we can conclude that: 1) our method outperforms the other methods with both the data sets; 2) the MF extended recommendation methods STE, SocialMF, and Graph Weighted Nonnegative Matrix Factorization (GWNMF) perform better than the Bounded Matrix Factorization (BMF) which purely uses the rating matrix for recommendations; 3) the trust methods TidalTrust and MoleTrust perform worse than the BMF, which shows that solely using trusted friends' opinions to recommend is not appropriate; and 4) the classic collaborative filtering methods UserCF and ItemCF perform worst of all methods,

TABLE III
COMPARISON RESULTS WITH ALL USERS

	Epinions			Flixster		
Methods	RMSE	Coverage	F-Measure	RMSE	Coverage	F-Measure
UserCF	1.3443	18.32%	0.2872	0.8562	64.82%	0.7105
ItemCF	1.3682	20.80%	0.3161	0.8447	68.56%	0.7336
TidalTrust	1.2524	37.84%	0.4880	0.8337	75.85%	0.7747
MoleTrust	1.2853	37.91%	0.4865	0.8243	76.73%	0.7804
BMF	1.2136	69.48%	0.6957	0.8148	87.13%	0.8321
STE	1.1724	78.52%	0.7440	0.8023	90.54%	0.8491
SocialMF	1.1228	82.15%	0.7670	0.7968	90.42%	0.8494
GWNMF	1.1045	85.67%	0.7847	0.7954	92.34%	0.8579
DLMF	**1.0736**	**87.64%**	**0.7975**	**0.7853**	**95.84%**	**0.8742**

TABLE IV
COMPARISON RESULTS WITH COLD-START USERS

	Epinions			Flixster		
Methods	RMSE	Coverage	F-Measure	RMSE	Coverage	F-Measure
UserCF	1.4658	12.57%	0.2098	0.9569	55.69%	0.6431
ItemCF	1.5433	13.26%	0.2181	0.9353	58.78%	0.6652
TidalTrust	1.3664	25.45%	0.3671	0.9268	68.24%	0.7228
MoleTrust	1.3719	27.18%	0.3845	0.9148	69.66%	0.7320
BMF	1.3258	54.36%	0.5996	0.8932	78.96%	0.7831
STE	1.2637	67.22%	0.6781	0.8556	85.35%	0.8184
SocialMF	1.2158	75.83%	0.7258	0.8273	86.37%	0.8269
GWNMF	1.1854	79.32%	0.7457	0.8194	88.62%	0.8382
DLMF	**1.1564**	**81.49%**	**0.7594**	**0.8043**	**92.46%**	**0.8572**

which indicates that when the data become extremely sparse, these methods are not applicable, since the similarity of users or items can be hardly achieved for the missing of most ratings. To sum up, among all these methods, our DLMF method can generally achieve a better performance on both precision and coverage. This demonstrates that our method for trust-aware recommendation is reasonable and efficient.

2) Comparison With Cold-Start Users: A major challenge of RSs is that it is difficult to recommend items to cold-start users, who have made quite few ratings. Users, who have expressed less than five ratings, are considered as cold-start users [26]. In the data set from Epinions, more than half of users are cold-start users. Therefore, it is significant to keep high efficiency for RSs to recommend for cold-start users. Hence, to compare our method with the others for cold-start users, we pick out the cold-start users in each test data and predict their ratings. Table IV shows that DLMF also outperforms other methods for cold-start users. We can also observe that the improvement of DLMF compared with the other methods for cold-start users is more than the improvement for all users which is introduced in Section V-E2. This indicates that DLMF can handle cold-start users better than other methods.

VI. CONCLUSION

Based on the intuition that users' decisions on the Internet items are affected by their both own characteristics and trusted friends' recommendations, we propose a novel MF method

for trust-aware recommendation by employing a deep learning technique. Our DLMF approach contains two phases of learning. In the first phase, we utilize a deep autoencoder to learn the initial values of the latent feature vectors of users and items. Then, in the second phase, we learn the final latent feature vectors of users and items, by minimizing the proposed objective function. The objective function not only includes the users' characteristics, but also the impact of the community effect in the trust social network. Besides, we propose a trust-aware algorithm for community detection. Our experimental analysis on the Epinions data set shows that our method outperforms the other state-of-the-art methods on recommendation accuracy.

In this paper, we consider the trust relationships of users in the social trust network being invariant. The trust relationship can change with time. Besides, the ratings from the users are also time sensitive, and ratings that are quite out-of-date may become noise information for recommendations. Therefore, we plan to involve the time sensitivity for trust-aware recommendation in our future work.

REFERENCES

[1] S. Deng, D. Wang, X. Li, and G. Xu, "Exploring user emotion in microblogs for music recommendation," *Expert Syst. Appl.*, vol. 42, no. 23, pp. 9284–9293, Dec. 2015.

[2] F. Alqadah, C. K. Reddy, J. Hu, and H. Alqadah, "Biclustering neighborhood-based collaborative filtering method for top-*n* recommender systems," *Knowl. Inf. Syst.*, vol. 44, no. 2, pp. 475–491, Aug. 2015.

[3] S. Deng, L. Huang, and G. Xu, "Social network-based service recommendation with trust enhancement," *Expert Syst. Appl.*, vol. 41, no. 18, pp. 8075–8084, Dec. 2014.

[4] A. Javari and M. Jalili, "A probabilistic model to resolve diversity–accuracy challenge of recommendation systems," *Knowl. Inf. Syst.*, vol. 44, no. 3, pp. 609–627, Sep. 2015.

[5] Y. Koren, "Factor in the neighbors: Scalable and accurate collaborative filtering," *ACM Trans. Knowl. Discovery Data*, vol. 4, no. 1, pp. 1–24, Jan. 2010.

[6] H. Zou, Z. Gong, N. Zhang, Q. Li, and Y. Rao, "Adaptive ensemble with trust networks and collaborative recommendations," *Knowl. Inf. Syst.*, vol. 44, no. 3, pp. 663–688, Sep. 2015.

[7] M. Jamali and M. Ester, "*TrustWalker*: A random walk model for combining trust-based and item-based recommendation," in *Proc. ACM Int. Conf. Knowl. Discovery Data Mining*, 2009, pp. 397–406.

[8] S.-G. Deng, L.-T. Huang, J. Wu, and Z.-H. Wu, "Trust-based personalized service recommendation: A network perspective," *J. Comput. Sci. Technol.*, vol. 29, no. 1, pp. 69–80, Jan. 2014.

[9] H. Ma, I. King, and M. R. Lyu, "Learning to recommend with explicit and implicit social relations," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 29–48, Apr. 2011.

[10] M. Jamali and M. Ester, "A matrix factorization technique with trust propagation for recommendation in social networks," in *Proc. ACM Conf. Recommender Syst.*, 2010, pp. 135–142.

[11] P. Avesani, P. Massa, and R. Tiella, "Moleskiing.it: A trust-aware recommender system for ski mountaineering," *Int. J. Infonomics*, vol. 20, no. 35, pp. 1–10, 2005.

[12] J. Golbeck, "Trust and nuanced profile similarity in online social networks," *ACM Trans. Web*, vol. 3, no. 4, pp. 1–33, Sep. 2009.

[13] J. A. Golbeck, "Computing and applying trust in Web-based social networks," Ph.D. dissertation, Dept. Comput. Sci., Univ. Maryland, College Park, College Park, MD, USA, 2005.

[14] P. Massa and P. Avesani, "Trust-aware recommender systems," in *Proc. ACM Conf. Recommender Syst.*, 2007, pp. 17–24.

[15] H.-F. Yu, C.-J. Hsieh, S. Si, and I. S. Dhillon, "Parallel matrix factorization for recommender systems," *Knowl. Inf. Syst.*, vol. 41, no. 3, pp. 793–819, Dec. 2014.

[16] R. Kannan, M. Ishteva, and H. Park, "Bounded matrix factorization for recommender system," *Knowl. Inf. Syst.*, vol. 39, no. 3, pp. 491–511, Jun. 2014.

[17] A. Mnih and R. Salakhutdinov, "Probabilistic matrix factorization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2007, pp. 1257–1264.

[18] P. Victor, M. Cock, and C. Cornelis, "Trust and recommendations," in *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. Kantor, Eds. New York, NY, USA: Springer, 2011.

[19] *Epinions*. [Online]. Available: http://www.epinions.com, accessed Jul. 10, 2014.

[20] P. Massa and P. Avesani, "Trust metrics in recommender systems," in *Computing With Social Trust*, J. Golbeck, Ed. London, U.K.: Springer, 2009.

[21] H. Abdi and L. J. Williams, "Principal component analysis," *Wiley Interdiscipl. Rev., Comput. Statist.*, vol. 2, no. 4, pp. 433–459, Jul./Aug. 2010.

[22] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[23] J. Wang, "Local tangent space alignment," in *Geometric Structure of High-Dimensional Data and Dimensionality Reduction*, J. Wang, Ed. Heidelberg, Germany: Springer, 2011.

[24] M. Belkin and P. Niyogi, "Laplacian Eigenmaps for dimensionality reduction and data representation," *Neural Comput.*, vol. 15, no. 6, pp. 1373–1396, Mar. 2003.

[25] M. Balasubramanian and E. L. Schwartz, "The Isomap algorithm and topological stability," *Science*, vol. 295, no. 5552, p. 7, Jan. 2002.

[26] P. Massa and P. Avesani, "Trust metrics on controversial users: Balancing between tyranny of the majority and echo chambers," *Int. J. Semantic Web Inf. Syst.*, vol. 3, no. 1, pp. 39–64, 2007.

[27] N. Guan, D. Tao, Z. Luo, and B. Yuan, "Online nonnegative matrix factorization with robust stochastic approximation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 7, pp. 1087–1099, Jul. 2012.

[28] A. Stuhlsatz, J. Lippel, and T. Zielke, "Feature extraction with deep neural networks by a generalized discriminant analysis," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 4, pp. 596–608, Apr. 2012.

[29] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[30] R. Salakhutdinov, A. Mnih, and G. Hinton, "Restricted Boltzmann machines for collaborative filtering," in *Proc. Int. Conf. Mach. Learn.*, 2007, pp. 791–798.

[31] K. Zhang, D. Lo, E.-P. Lim, and P. K. Prasetyo, "Mining indirect antagonistic communities from social interactions," *Knowl. Inf. Syst.*, vol. 35, no. 3, pp. 553–583, Jun. 2013.

[32] S. Fortunato, "Community detection in graphs," *Phys. Rep.*, vol. 486, nos. 3–5, pp. 75–174, 2010.

[33] Z. Zhou, W. Wang, and L. Wang, "Community detection based on an improved modularity," in *Pattern Recognition*. Berlin, Germany: Springer, 2012.

[34] R. Zdunek, "Initialization of nonnegative matrix factorization with vertices of convex polytope," in *Artificial Intelligence and Soft Computing*. Berlin, Germany: Springer, 2012.

[35] B. Hidasi and D. Tikk, "Initializing matrix factorization methods on implicit feedback databases," *J. Universal Comput. Sci.*, vol. 19, no. 12, pp. 1834–1853, 2013.

[36] X. Yang, H. Steck, and Y. Liu, "Circle-based recommendation in online social networks," in *Proc. ACM Int. Conf. Knowl. Discovery Data Mining*, 2012, pp. 1267–1275.

[37] C. H. Nguyen and H. Mamitsuka, "Latent feature kernels for link prediction on sparse graphs," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 11, pp. 1793–1804, Nov. 2012.

[38] M. Rezaei, R. Boostani, and M. Rezaei, "An efficient initialization method for nonnegative matrix factorization," *J. Appl. Sci.*, vol. 11, no. 2, pp. 354–359, 2011.

[39] H. Zhang, Z. Yang, and E. Oja, "Improving cluster analysis by co-initializations," *Pattern Recognit. Lett.*, vol. 45, no. 1, pp. 71–77, Aug. 2014.

[40] Q. Gu, J. Zhou, and C. Ding, "Collaborative filtering: Weighted nonnegative matrix factorization incorporating user and item graphs," in *Proc. SIAM Int. Conf. Data Mining*, 2010, pp. 199–210.

Shuiguang Deng (M'14) received the B.S. and Ph.D. degrees in computer science from Zhejiang University, Hangzhou, China, in 2002 and 2007, respectively.

He was a Visiting Scholar with the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2014, and Stanford University, Stanford, CA, USA, in 2015. He is currently an Associate Professor with the College of Computer Science and Technology, Zhejiang University. His current research interests include service computing and business process management.

Longtao Huang received the B.S. and Ph.D. degrees in computer science from Zhejiang University, Hangzhou, China, in 2010 and 2015, respectively.

He is currently a Research Staff Member with the Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China. His current research interests include service computing and cloud computing. He is the corresponding author of this paper.

Guandong Xu (M'05) received the Ph.D. degree in computer science from Victoria University, Melbourne, VIC, Australia.

He is currently a Lecturer and an Analytic Education Program Leader with the Advanced Analytics Institute, University of Technology at Sydney, Sydney, NSW, Australia. He has authored three monographs with the Springer and the CRC Press, and dozens of journal and conference papers. His current research interests include Web mining and Web search, data mining and machine learning, recommender systems, and social network analysis.

Dr. Xu has served on the Editorial Board or as a Guest Editor for several international journals, such as the *Computer Journal*, the *Journal of Systems and Software*, and the *World Wide Web Journal*. He is the Assistant Editor-in-Chief of the *World Wide Web Journal*.

Xindong Wu (F'11) received the bachelor's and master's degrees in computer science from the Hefei University of Technology, Hefei, China, and the Ph.D. degree in artificial intelligence from the University of Edinburgh, Edinburgh, U.K.

He is currently a Professor of Computer Science with The University of Vermont, Burlington, VT, USA. His current research interests include data mining, knowledge-based systems, and Web information exploration.

Prof. Wu is a fellow of the American Association for the Advancement of Science. He is the Steering Committee Chair of the IEEE International Conference on Data Mining, and the Editor-in-Chief of *Knowledge and Information Systems*. He was the Editor-in-Chief of the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING from 2005 to 2008.

Zhaohui Wu (SM'04) received the Ph.D. degree from Zhejiang University, Hangzhou, China, in 1993.

He was with the German Research Center for Artificial Intelligence, Bremen, Germany, as a Joint Ph.D. Student from 1991 to 1993. He has been involved in research on service computing and intelligent systems for a long time. He is currently the President of Zhejiang University and the Director of the Institute of Computer System and Architectures. He has authored over 180 papers and nine books, and established a series of international conferences, including the International Conference on Electron Spectroscopy and Structure, the International Conference on Cyber, Physical and Social Computing, and MSCI.