Collaborative Filtering using a Trust Social Network with Deep Learning for Initialization

Weijia Zhang
Department of Statistics
University of Washington
weijia94@uw.edu

Reed Zhang
Department of Statistics
University of Washington
rzhang1@uw.edu

Zhitao Yu
Department of ME, Statistics
University of Washington
zhitaoyu@uw.edu

Abstract

In Recommender Systems, collaborative filtering is a technique that uses ratings provided by users to suggest relevant items. Although commonly used in many well-known online commercial systems (e.g. Amazon[1], Netflix[2]), collaborative filtering suffers from problems of data sparsity and cold start users. In respond to these challenges, we propose a novel algorithm that utilizes user-item rating information as well as the user-user trust network. Most social recommendation models identify an active user's connections and estimate his rating on an unknown item as a weighted sum of ratings provided by connected users. The weights are estimated by either user similarities or connection degrees. However, this type of mechanism fails to generate accurate recommendations for users who do not disclose any social relationships. Thus, most social-aware recommender models still face the sparsity issue when there are isolated users in the social network. In this paper, we investigate ways to leverage community detection on a trust network to circumvent this sparsity issue. In particular, we improve the community detection procedure, *TrustCliques*, to accommodate for users with no trust relationships. Our approach also uses matrix factorization to learn latent representations for users and items. Deep learning is used to search for a better initialization in the non-convex optimization problem. Experimental results on the Epinions dataset demonstrate that accounting for users with zero trust improves model recommendations.

1 Introduction

Recommender Systems (RS) are algorithms widely used to suggest relevant items to users. There are two major paradigms of RS: collaborative filtering (CF) and content-based methods. While content-based RS requires description of the characteristics of the item, CF builds upon the rating history provided by users; therefore, CF techniques are often used when descriptions of items are unavailable. The core assumption of CF is that users who acted similarly in the past will share similar preference in the future. A typical CF algorithm first assesses similarity between an active user and other users, and predicts the rating the active user would give to a certain item. The prediction is the weighted sum of the ratings provided by other users on the item, and weights are the weighted user similarities.

CF suffers some inherent weaknesses that are innate in the process of finding similar users:

1. *Data sparsity:* In order to quality similarity between two users, they should have rated at least a few items in common. However, given the large item sets (e.g. products on Amazon, shows on Netflix), most users only have rated a tiny percentages of the items. As

[1] https://www.amazon.com/
[2] https://www.netflix.com/

a consequence, the system has to choose neighbors within a small portion of comparable users and is likely to miss other non-comparable but relevant users [5].

2. *Cold start users:* This is a group of users who provided few or none ratings. For this group of users, it is often not possible to compare them with other users and to find their neighbors. Therefore, CF often fails to generate recommendations to cold start users.

3. *Unreliable users:* Conventional CF mechanism has difficulty to detect unreliable/"malicious" users. For example, imagine you are a movie producer and you want to trick Netflix to always recommend the movies you produced. It is surprisingly not difficult to attack the Web if it employs CF. O'Mahony *et al.* [6] explained different attack types to achieve this goal, the simplest one is the *copy-profile attack*: the attacker copies most ratings of a target user to fool the system into identifying the attacker as the most similar user to the target, hence every additional item that the attacker gives a good review on will be recommended to the active user.

4. *Neglect of social network:* Traditional CF assumes users are *i.i.d.* (independent and identically distributed), hence fails to recognize the impact on a target user from other users in her social network. It is intuitive to understand that the users are more likely to accept recommendations from their friends, families, or people they trust. However, regular CF can not utilize this influence.

Therefore, the traditional CF work, which purely mines the user-item-rating data set, fails to provide accurate and reliable recommendations. In response to these challenges, we propose a novel CF work that employs both user-item-rating information and users' social web information. An important step in our method is to mine the web of social connections to form communities. For a target user, our model considers rating history of users in her communities when predicting her review on an unknown item. The community detection method we employ in this paper encourages each user to be classified into multiple communities, and each community consists of at least two users. Therefore, our method mitigates the issues with data sparsity and cold start users. Also, the communities are built based on social connections disclosed by users themselves, which makes it more difficult for unreliable users to attack the system. Moreover, by taking the social network into account, our model clearly circumvents the last weakness mentioned above.

There are two types of social relation, directed and undirected. Friend on Facebook[3] is an example of undirected relation and follow on Instagram[4] is an example of directed relation. In this paper we assume the social web is formed by *trust*, a type of directed relation. However, one can easily extend the proposed method to the other case.

There are two important questions to consider when building a trust based CF system: (1) how to decide the significance of each trust relation; (2) how to make predictions for users who have zero trust network, that is, users who do not disclose his trust and are trusted by no one. In this paper, we user *nonconformists* to describe a user who has no trust connections. We make the following assumptions in our method.

1. Trusts placed by a user have different significances, and the entire trust network should be considered when measuring such significances.

2. The existence of nonconformists does not necessarily indicate a lack of trust; another explanation is that some users choose not to disclose their trust when reading review.

In Section 2, we show that most previous works on trust CF violate the two assumptions, which makes our method different from them. The rest of the paper is organized as follows. Section 2 presents some related work. Section 3 formally defines the recommendation problem we are trying to solve in this paper. The detailed explanation of our method is in Section 4. In section 5, we provide our experimental design and discuss the empirical results. Section 6 is the conclusion and future work.

[3] `https://www.facebook.com/`

[4] `https://www.instagram.com/`

2 Related Work

Trust-aware recommendation is widely studied since it provides an alternative way to study user preferences. Guo *et al.* [2] suggested *TrustSVD*, a model built upon *SVD++* [3]. To incorporate a trust network in recommendations, Guo *et al.* suggested using a latent-space representation to factor the trust matrix and then scaling by the square root of the number of users trusted by a user. Although it was shown in [2] that *TrustSVD* outperforms other state-of-art CF algorithms on several datasets, further improvements can be made for better performance. One underlying assumption in *TrustSVD* is that the trustees of a user influence that user in a uniform manner. This assumption ignores the impact of influencers, users who are trusted by many other users, who may have a disproportionate amount of influence on those who trust them.

Ma *et al.* [4] made the same assumption in their proposed model *SoRec*. Their mechanism is to weight trust effects based on in-degrees and out-degrees of each user. The intuition behind this is that the confidence of a trust value should be decreased if a user trusts many users, and increased if a user is trusted by many users [4]. However, this mechanism only considers local user influence and fails to recognize the true user influence value in the global network. Moreover, *SoRec* only looks at first-degree trust connections which makes the model still suffer from sparsity. Additionally, *SoRec* assumes independence between trust connections, which is unrealistic in trust propagation.

Tang *et al.* [7] proposed a model, *SoDimRec*, that is able to capture heterogeneity in social relationships and incorporate trust information beyond just first-degree connections. Moreover, *SoDimRec* relaxes the assumption of independence in the trust network by forming communities for each active user. In their study, heterogeneity refers to the idea that different trustees may influence a user in different ways. They first performed community detection on the trust graph to identify groups of users who frequently interact with each other, and associated with each group a preference vector. They then added two additional terms to the standard MF objective function: one to encourage the group preference vectors to be close to their users' preferences, and another to encourage weakly-connected users within each group to have similar preferences to each other. Weakly-connected users are defined as users in the same group who do not have a direct trust connection. *SoDimRec* is able to capture more complex relationships in the trust graph, offering improvements to *SoRec* and *TrustSVD*. However, *SoDimRec*, *SoRec*, and *TrustSVD* do not provide any improvements for users without trust relationships. These algorithms suffer when there is sparsity in the trust network, i.e. when many active users are nonconformists.

Because optimization in MF approaches can converge to different local minima, a good setting of the initialization values is crucial for getting near-optimal results. In most MF approaches, the solutions for latent representation of users and items are randomly initialized. Shuiguang *et al.* [1] used deep learning for pretraining to obtain better initializations. They proposed an auto-encoder neural network method to learn a compressed representation for the user ratings, using a continuous Restricted Boltzmann Machines (CRBMs) for pretraining weights. They also incorporated trust in the model and considered community effects, similarly to *SoDimRec*. Although the time cost for initialization was greater using the autoencoder, their approach performed better than random initialization. However, this time cost is only relevant during the training stage, so it should not influence the speed of using the recommender system.

3 Problem Definition

The recommendation problem in this paper is to predict the rating that a user will give on an unknown item, using the user-item rating matrix and the user-user trust matrix. Suppose there are m users and n items in the system. Let $R = [r_{u,i}]_{m \times n}$ denote the user-item rating matrix where each $r_{u,i}$ represents the rating on item i given by user u. We employ low-rank Matrix Factorization (MF) to find two matrices: a user latent feature matrix $P = [p_1, ..., p_m]^T \in \mathbb{R}^{m \times k}$ and an item latent feature matrix $Q = [q_1, ..., q_n]^T \in \mathbb{R}^{n \times k}$, where p_u and q_i are low-dimensional latent representations of user u and item i. Under the MF framework, R is estimated by the product PQ^T; *i.e.*, $\hat{r}_{u,i} = p_u^T q_i$. The latent matrices P, Q are then obtained by solving the minimization problem in (1).

$$\mathcal{L} = \frac{1}{2} \min_{P,Q} \sum_{u=1}^{m} \sum_{i=1}^{n} I_{u,i} (r_{u,i} - \hat{r}_{u,i})^2 + \frac{\lambda_1}{2} \|P\|_F^2 + \frac{\lambda_2}{2} \|Q\|_F^2 , \tag{1}$$

where $I_{u,i}$ is an indicator function such that $I_{u,i} = 1$ if user u has an explicit rating for item i, and 0 otherwise; $\|\cdot\|_F$ is the Frobenius norm.

In addition to R, suppose we have a directed social network graph $G = (\mathcal{V}, \mathcal{E})$, where the vertex set $\mathcal{V}$ represents the set of m users and the edge set $\mathcal{E}$ indicates the directed trust relationship among users. We use $T = [t_{u,v}]_{m \times m}$ to denote the adjacency matrix describing trust propagation among users; $t_{u,v} = 1$ indicates u trusts v and 0 otherwise. Note that trust is a directed relationship, hence T is generally asymmetric.

4 Trust-aware Recommendation

In this section we present the details of our trust based CF algorithm. In our approach, we employ the deep autoencoder proposed by Deng *et al.* to learn initial values for latent feature matrices of users and items. We use communities to reflect social dimensions in the trust network, and we consider the influence of communities that a user belongs to on his predicted rating on an unknown item. In addition, our algorithm allows various degrees of influence based on users' social value in the trust network.

4.1 Deep Learning Initialization

Following the techniques in [1], we use an autoencoder to learn a compressed representation for the user ratings, with continuous Restricted Boltzmann Machines (CRBMs) for pretraining weights. The deep autoencoder starts with learned weights from the CRBMs and is then trained via back propagation on reconstruction error.

Define the user matrix $U_{n \times m} = R^T$ and item matrix $I_{m \times n} = R$. The deep autoencoder can be used to get initial values for P and Q by extracting features from the original user/item matrices U and I. From an input $U_{n \times m}$ to the network, the output of one CRBM is $O_{p \times m}$. (In the deep autoencoder, there are multiple CRBMs). There are n visible units and p hidden units. For each visible unit v_i and hidden unit h_j, w_{ij} is the weight. The encoding stage proceeds as follows:

1. Compute the hidden units via:

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

   where $N_j(0,1)$ is a standard normal random variable; $n_j = \sigma N_j(0,1)$ is the noise input component, which has standard deviation σ. $\phi_j(x)$ are the asymptotes at θ_L and θ_H in the equation

$$\phi_j(x) = \theta_L + (\theta_H - \theta_L) \frac{1}{1 + exp(-a_j x)},$$

   where a_j is the parameter to influence the slope of the sigmoid function.

2. Reconstruct each visible vector v according to:

$$v_i' = \phi_i \left(\sum_j w_{ji} h_j + \sigma N_i(0,1) \right)$$

3. Recompute the hidden units h' using the equations in stage 1.

4. Update the weights w_{ij} and the noise controller a_j by minimizing the contrastive divergence:

$$\Delta w_{ij} = \eta_w (v_i h_j - v_i^{'} h_j^{'})$$
$$\Delta a_{ij} = \frac{\eta_a}{a_j^2} (h_j^2 - {h_j^{'}}^2)$$

There are multiple CRBMs in this deep autoencoder. The input data of each CRBMs is the output of the previous CRBMs. The fine tune stage uses backpropagation to minimize the l_2 reconstruction error $\sqrt{\sum_i (v_i - \hat{v}_i^2)}$.

4.2 Influence Ranking

In our model we take users' social influence into account when making predictions on an unknown rating. In online social networks, users exhibit different degrees of influence, and such information is valuable in predicting unknown ratings. Many trust-aware RS models overlook the importance of user influence in the network and only consider pairwise user similarities. Among the algorithms we introduced in Section 2, only Ma *et al.* utilized user influence in their model, *SoRec*. However, they model the social influence of a target user u using only the number of users u trusted, and the number of users who trusted u. We believe that their method does not reflect true social influence as the significance of trust varies based on user characteristics. For example, trust placed by influential users, users who are trusted by many others, should be considered as more credible compare to trust placed by users who are trusted by few. Therefore, a user's influence should be modeled by a recursive method that recognizes both the degree of incoming trust received and the importance of each connection. For these two specific reasons, we utilize *PageRank* to estimate social influence for each user. *PageRank* is a link analysis algorithm that assigns a numerical value to each node in a graph; such value can be viewed as a level of importance in the graph network.

4.3 Community Detection

In this section we present a community detection (CD) algorithm that embraces the CD technique *TrustCliques* proposed by Deng *et al.*. We also describe our strategy for allocating nonconformists. *TrustCliques* generates d-cliques, subgraphs where the largest distance between any pair of nodes in the clique is at most d. The generated communities by our CD algorithm are inline with three assumptions. First, the mapping from users to communities is one-to-many. This assumption is based on the intuition that a user can belong to different communities based on different interests. Second, users within the same community form a denser trust network than users across different communities. Third, to circumvent the sparsity issue, each community consists of at least two users.

The CD algorithm *TrustCliques* is a greedy method that updates community structure based on maximum gain on the parameter *collectivity*. The definition of collectivity is as follows.

Definition 4.1 (*Collectivity*)**.** Given a trust adjancency matrix T and the corresponding trust network graph $\mathcal{G}$, assume that the users have been clustered into a set of cliques C_i. The collectivity *col* is defined as:

$$\text{col} = \log\left(\frac{\sum_{C_i \in \text{clqs}} \text{col}_i}{|\text{clqs}|}\right) \quad \text{with} \quad \text{col}_i = \frac{\sum_{u,v \in C_i} t_{u,v}}{\sum_{u \in C_i, v \notin C_i} t_{u,v}},$$

where *clqs* is the set of cliques.

We chose *TrustCliques* as our base CD technique for three reasons. First, *TrustCliques* detects overlapping communities, which are inline with our first assumption. Second, *TrustCliques* is able to form small and highly connected communities. Third, *TrustCliques* is a simulated annealing method so that the algorithm has a better chance to reach the global optimum.

While *TrustCliques* forms communities to reflect our first two assumptions, the third assumption (no communities of size one) is violated. This violation leads to sparsity issue and we fix it by merging similar communities. Let $C = \{C_1, ..., C_c\}$ be the set of communities detected by *TrustCliques*. We use $\hat{p}_i$ to denote the latent feature factor for the community C_i, and let $\hat{P} = [\hat{p}_1, ..., \hat{p}_c]^T \in \mathbb{R}^{c \times k}$. Since a user can be classified into multiple communities, she might have different degrees of association to different communities. To model such association, let $S_{u,k}$ denote the strength of user u associating with community C_k. Inspired by *SoDimRec*, we define $S_{u,k}$ as the following,

$$S_{u,k} = \beta_1 \frac{\sum_{v \in C_k} \text{sim}(u,v)}{|C_k|} + \beta_2 \frac{|\mathcal{N}_u \cap C_k|}{|\mathcal{N}_u|} + (1 - \beta_1 - \beta_2) \cdot \textit{PageRank}_u,$$

where $\mathcal{N}_u = \{v | t_{u,v} = 1 \text{ or } t_{v,u} = 1\}$ and $\text{sim}(u,v)$ is the cosine similarity of user u and user v. The strength of $S_{u,k}$ is defined as a linear combination of the user similarity, the overlap between u's social network and community C_k, and the influence ranking of u. $\hat{P}$ is obtained by minimizing the

objective in (2).

$$\mathcal{L} = \min_{P,Q,\hat{P}} \frac{1}{2} \sum_{u=1}^{m} \sum_{i=1}^{n} I_{u,i}(r_{u,i} - \hat{r}_{u,i})^2 + \frac{1}{2} \sum_{u=1}^{m} \sum_{k=1}^{c} \tilde{I}_{u,k} S_{u,k} \left\| p_u - \hat{p}_k \right\|_2^2 + \frac{\alpha}{2}(||P||_F^2 + ||Q||_F^2 + ||\hat{P}||_F^2), \tag{2}$$

where $\tilde{I}_{u,k}$ is an indicator function that takes value 1 if user u is in community C_k and 0 otherwise. The first term of (2) is the sum of squared differences between true and predicted ratings. The second term encourages latent-space representations of communities to be close to the latent-space representations of their users. The third term prevents overfitting.

Once obtained, $\hat{p}_k$ is treated as the fixed latent-space representation for community C_k. By interpreting nonconformists as communities of size one, we can measure the similarity between any nonconformist and a given community C_k. Each nonconformist is then assigned to the f communities to which they are the most similar. After assigning each nonconformist, we have a set of communities that are in line with our three assumptions. The detailed implementation of the described CD technique is in Algorithm 1.

Algorithm 1 Community Detection

Input: maximum clique size d, iteration limitation l, user set U, trust matrix $\hat{T}$
Output: cliques collection *clqs*

clqs=U, $n_c = |clqs|$
while $n_c > 1$ and number of iteratinos $< l$ **do**
  for $i = 1$ to n_c **do**
    C_i = the i-th clique in *clqs*;
    J = the set of immediate neighbors of C_i;
    $\Delta\text{col}_{\text{max}} = 0$;
    for each $v \in J$ **do**
      $C_i' = C_i \cup v$;
      if C_i' is a d-clique **then** calculate Δcol
        if $\Delta\text{col} > \Delta\text{col}_{\text{max}}$ **then**
          $\Delta\text{col}_{\text{max}} = \Delta\text{col}$;
          $C_t = C_i'$;
        end if
      end if
    end for
    if $\Delta\text{col}_{\text{max}} > 0$ or $U(0,1) > prob$ **then**
      replace C_i with C_t in *clqs*;
      store current *clqs* and the corresponding collectivity
    end if
  end for
  $n_c = |clqs|$, iteration step ++;
end while
C is the set of communities of size at least two; ▷ nonconformists allocation
$\tilde{U}$ is the set of nonconformists
compute $\hat{P}$ and P by solving the minimization problem in (2);
for each $\tilde{u} \in \tilde{U}$ **do**
  for each $\hat{p}_k \in \hat{P}$ **do**
    $\text{sim}_{\tilde{u},k}$ = cosine similarity between $p_{\tilde{u}}$ and $\hat{p}_k$;
  end for
  merge $\tilde{u}$ into f communities in C based on the top f sim scores;
end for
return the set of communities $\mathcal{C} = \{\mathcal{C}_1, ..., \mathcal{C}_c\}$

4.4 Main Objective

Our framework tries to minimize the objective in equation (2) using gradient descent with backtracking. The optimization procedure is run twice: first (2) is optimized over P, Q, and $\hat{P}$. As noted in Algorithm 1 (line 23-30), $\hat{P}$ is used to allocate nonconformists to communities. After the allocation, $\hat{P}$ is treated as fixed, while P and Q are further updated with the new communities. The gradients of p_u, q_i, and $\hat{p}_k$ are

$$\frac{\partial \mathcal{L}}{\partial p_u} = \sum_{i=1}^{n} I_{u,i}(r_{u,i} - \hat{r}_{u,i})q_i + \sum_{k} \tilde{I}_{u,k} S_{u,k}(p_u - \hat{p}_k) + \alpha p_u$$

$$\frac{\partial \mathcal{L}}{\partial q_i} = \sum_{]u=1}^{n} I_{u,i}(r_{u,i} - \hat{r}_{u,i})p_u + \alpha q_i$$

$$\frac{\partial \mathcal{L}}{\partial \hat{p}_k} = \sum_{u=1}^{m} \tilde{I}_{u,k} S_{u,k}(\hat{p}_k - p_u) + \alpha \hat{p}_k$$

5 Experiments and Results

5.1 Data

We choose Epinions[5] dataset to test the empirical performance of our model. This dataset is provided by Tang *et al.*. Readers can download the .mat and the .txt versions of the dataset on the following site:

`https://www.cse.msu.edu/~tangjili/datasetcode/truststudy.htm`.

Epinions.com is a consumer review site that was established in 1999. The dataset contains ratings and trust lists for each user, as well as product ID, category name, time of creating, and the helpfulness for rating record. The statistics of the dataset are in Table 1. We propose to use this dataset for two reasons. First, it is relatively sparse compared to other commonly used datasets for CF work. The rating density in the Epinion dataset is 0.014%, whereas the densities of MovieLens and Eachmovie (two famous CF datasets) are 4.25% and 2.29%, respectively [4]. Therefore, we can examine how well our model handles sparsity. Second, a trust social network is included in the Epinion dataset.

users	items	ratings	density	trusters	trustees	relations	density
22,164	296,277	922,267	0.014%	15,448	15,867	355,813	0.145%

Table 1: Statistics of Epinion dataset

5.2 Experimental Setting

We compare our method (*TrustCF+NC+DLI*) with the following models to test its performance:

1. *UserCF*: user-based collaborative filtering method
2. *itemCF*: item-based collaborative filtering method
3. *TrustCF*: method described in this paper but without nonconformist allocation or the deep learning initialization
4. *TrustCF+NC*: method described in this paper but without the deep learning initialization

The following parameter set is used in all models:

$$\{k = 80, \beta_1 = 0.6, \beta_2 = 0.2, \alpha = 0.1, f = 3\}.$$

We ran *TrustCliques* to search for 2-cliques until global collectivity was increasing by less than 0.25% per iteration. The algorithm found 12549 communities with an average size of 6.80. There are 4,076 nonconformists (roughly 18.4% of the total population). After we allocate these nonconformists as described in Algorithm 1, the average community size is 7.78.

[5] `https://epinionglobal.com/en/`

5.3 Evaluation

We use two metrics, mean absolute error (MAE) and root mean square error (RMSE) to evaluate predictions.

$$\text{MAE} = \frac{\sum_{u,i}|\hat{r}_{u,i} - r_{u,i}|}{N}, \qquad \text{RMSE} = \frac{\sum_{u,i}(\hat{r}_{u,i} - r_{u,i})^2}{N},$$

where N is the number of test ratings. We randomly split the data into an 80-20 train-test split, evaluating MAE and RMSE on the test set.

5.4 Empirical Results

The experimental results are shown in Table 2 for RMSE and MAE. As we can see from the table, *TrustCF+NC* outperforms the other models. Compared to *userCF* and *itemCF*, *TrustCF+NC* decreases the RMSEs by 6.5% and 5.5%, respectively. The RMSEs and MAEs of *TrustCF* and *TrustCF+NC* indicate an improvement on prediction accuracy after nonconformists allocation.

Metric	*userCF*	*itemCF*	*TrustCF*	*TrustCF+NC*	*TrustCF+NC+DLI*
RMSE	1.14238	1.12959	1.08626	**1.06756**	1.18136
MAE	0.84662	0.83897	0.81595	**0.81101**	0.87001

Table 2: Comparison Results

From the last column, we see that the model performed worse with the deep learning initialization, even when compared to *userCF* or *itemCF*. This indicates possible improvements in the Deep Learning pretraining process and can be investigated in future work. The focus of this paper is to study CF with a trust network, especially with a strategy for nonconformist allocation, and Table 2 suggests that allocating nonconformists to similar communities yields better predictions.

We want to note that although *TrustSVD*, *SoRec*, and *SoDimRec* used the Epinions dataset in [2],[4], and [7], they used different versions of the dataset. For example, The Epinions dataset used in the *SoDimRec* paper has 21,882 users, 59,104 items, and 632,663 ratings, which is smaller than the data we used for this paper. We were unable to find the exact version of the data to compare our results with previous methods. For reference, RMSs reported by the *TrustSVD*, *SoRec*, and *SoDimRec* in the original papers were 1.044, 1.1069, and 1.1811, respectively.

6 Conclusions

In this paper we presented our proposal for improving a trust based CF system to provide more accurate predictions. The social recommendation problem has been studied in the literature before. However, to our best knowledge, none have considered methods to deal with nonconformists, i.e. users who lack trust connections. In most state-of-art trust based CF systems, nonconformists are isolated nodes in the social graph, so they are interpreted as users who have no influence in the Web of trust. We believe this interpretation is misleading in situations when users are simply reluctant to disclose their social relations. Therefore, we propose our method to accommodate for nonconformists. The empirical results on the real-world data set Epinions suggests that properly handling nonconformists can yield better predictions.

Community structure is an important aspect in our model, as we believe that heterogeneity in social relationships is better captured by forming communities. We make three assumptions about communities: overlapping, dense (in terms of trust), and at least of size two. By satisfying these assumptions, the proposed model is able to (1) mitigate the sparsity issue; (2) reflect the fact that people may turn to different groups for different opinion; (3) consider impact from a user's direct and indirect trustees.

6.1 Future Work

In this section we discuss potential work to further investigate trust-based recommendation models.

1. *Hybrid RS using trust network:* given the promising results in this paper, we believe a hybrid RS model could be promising as well. Hybrid RS are mix of CF and content-based methods, and are often used as an upgraded RS mechanism.
2. *Content based influence measure:* in this paper we emphasize the fact that a user may turn to different groups for recommendations based on different interests. For example, user u may prefer v's recommendation on books but trust w more when it comes to movies. Therefore, another possible improvement could be to measure user influence within communities instead of globally.
3. *The deep learning initialization:* In this paper, we implement a deep auto-encoder to get the initialization of the latent factor. Unfortunately it does not behave as we expected. The reason might be because the hyperparameters of the deep auto-encoder are not fine tuned. Also, we can apply other state of art deep learning initialization methods such as variational auto-encoder or Generative Adversarial Networks (GANs).

6.2 Individual Contribution

Weijia Zhang: Proposed research topic and dataset. Proposed and implemented nonconformist allocation and user influence ranking. Defined metrics, worked on data preliminary analysis and report-writing (except for Section 4.1 and some of Section 2).

Reed Zhang: Implemented community detection, user-community similarity, and gradient descent with backtracking for sparse matrix representations to minimize the objective. (Up to line 25 in Algorithm 1.) Revised report for clarity.

Zhitao Yu: Proposed and implemented deep learning initialization algorithm (Deep autoencoder) for sparse matrix representations, including weight initialization, relu activation, backpropagation and reconstruction error, etc. Worked on report writing.

References

[1] Shuiguang Deng, Longtao Huang, Guandong Xu, Xindong Wu, and Zhaohui Wu. On deep learning for trust-aware recommendations in social networks. volume 28, pages 1164–1177. IEEE, 2016.

[2] Guibing Guo, Jie Zhang, and Neil Yorke-Smith. Trustsvd: Collaborative filtering with both the explicit and implicit influence of user trust and of item ratings. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[3] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434, 2008.

[4] Hao Ma, Haixuan Yang, Michael R Lyu, and Irwin King. Sorec: social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 931–940, 2008.

[5] Paolo Massa and Paolo Avesani. Trust-aware collaborative filtering for recommender systems. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 492–508. Springer, 2004.

[6] Michael P O'Mahony, Neil J Hurley, and Guénolé CM Silvestre. Recommender systems: Attack types and strategies. In *AAAI*, pages 334–339, 2005.

[7] Jiliang Tang, Suhang Wang, Xia Hu, Dawei Yin, Yingzhou Bi, Yi Chang, and Huan Liu. Recommendation with social dimensions. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*, pages 251–257, 2016.