Robust Preference-Guided Denoising for Graph based Social Recommendation

Yuhan Quan  
Department of Electronic Engineering, Tsinghua University  
Beijing, China

Jingtao Ding∗  
Department of Electronic Engineering, Tsinghua University  
Beijing, China

Chen Gao  
Department of Electronic Engineering, Tsinghua University  
Beijing, China

Lingling Yi  
Wechat, Tencent  
Shenzhen, China

Depeng Jin  
Department of Electronic Engineering, Tsinghua University  
Beijing, China

Yong Li  
Department of Electronic Engineering, Tsinghua University  
Beijing, China

ABSTRACT

Graph Neural Network (GNN) based social recommendation models improve the prediction accuracy of user preference by leveraging GNN in exploiting preference similarity contained in social relations. However, in terms of both effectiveness and efficiency of recommendation, a large portion of social relations can be redundant or even noisy, e.g., it is quite normal that friends share no preference in a certain domain. Existing models do not fully solve this problem of relation redundancy and noise, as they directly characterize social influence over the full social network. In this paper, we instead propose to improve graph based social recommendation by only retaining the informative social relations to ensure an efficient and effective influence diffusion, i.e., graph denoising. Our designed denoising method is preference-guided to model social relation confidence and benefits user preference learning in return by providing a denoised but more informative social graph for recommendation models. Moreover, to avoid interference of noisy social relations, it designs a self-correcting curriculum learning module and an adaptive denoising strategy, both favoring highly-confident samples. Experimental results on three public datasets demonstrate its consistent capability of improving two state-of-the-art social recommendation models by robustly removing 10–40% of original relations. We release the source code at https://github.com/tsinghua-fib-lab/Graph-Denoising-SocialRec.

KEYWORDS

Social Recommendation, Graph Denoising, Preference Learning

ACM Reference Format:


1 INTRODUCTION

Nowadays, social recommendation has become an important scenario of personalized recommender systems, where both user-item interactions and user-user social connections are available in the platforms [40]. Previous studies on human behaviors have established two basic theories describing the impact of social relations between users: 1) socially-connected users tend to possess similar preferences, which is referred as social homophily [33], and 2) behavior of a user can be influenced by her friends, like co-purchasing the same item, which is referred as social influence [32]. Above theories regarding the impacts of social relations on recommendation have been validated by recent applications in areas like advertising [2], e-commerce [6, 54] and etc.

To characterize the above social relation effects on user preference, researchers have developed many social recommendation methods, like achieving preference similarity with social regularization [31], enhancing influence from trust relations [18], and exploiting different relations with strong or weak ties [46]. Recent progress of graph neural network (GNN) in graph machine learning (ML) has further boosted the development of graph based social recommendation [15], as messaging passing mechanism widely adopted in GNN is well suited for characterizing influence diffusion in social context [13, 41, 51]. Moreover, aided by self-supervised learning techniques, GNN based social recommendation (short for GSocRec) models can extract more informative signals from massive observed social relations [58, 60].

However, contrastive to the prevalence of GSocRec models, one important problem that has been barely explored is the existence of redundant or even noisy social relations in the context of graph based social recommendation. As we have shown in Figure 1(a), most users only share similar preferences with a small portion of friends. Specifically, the median ratio value of friends having co-interactions is about 30% and 20% in two empirical datasets, respectively. Therefore, the rest social relations may be redundant or even harmful for social recommendation, as it is quite possible that two friends have distinct preferences in some areas. Consequently, previous GSocRec models that directly characterize social influence over the full social network have two shortcomings. One is the huge burden on both computation and storage brought by learning
The core of this framework is a curriculum learning module and an adaptive denoising strategy explicitly co-supervised by a recommendation loss. As for the second challenge, the first challenge of characterizing relation confidence, GDMSR is designed to better exploit preference guidance in a two-fold manner, i.e., a robust preference-guided graph denoising capability, GDMSR can achieve 10-40% of reduction ratio on social graphs without hurting recommendation accuracy, as well as maintaining high practicality in terms of efficiency and scalability.

2 PRELIMINARIES

2.1 GNN based Social Recommendation Model

Generally, a typical GNN based social recommendation (short for GSocRec) model involves two key parts of social influence diffusion and user preference learning. Specifically, the former characterizes the impact of friends’ preferences on a certain user, while the latter stands for training a prediction model based on user-item interactions, as a common step in recommender systems. Mathematically, a social network of $N$ users is denoted as a user-user graph $G=(\mathcal{U}, \mathcal{E})$, where $\mathcal{U} = \{u\}$ and $\mathcal{R} = \{(u, v)|r_{uv} = 1, \forall u, v \in \mathcal{U}\}$ denote the set of users and social relations, respectively. Besides, there is another bipartite graph $G_r=(\mathcal{U} \cup \mathcal{M}, \mathcal{P})$ that stands for the interactions between $N$ users $\mathcal{U}$ and $M$ items $\mathcal{M}$. To represent users and items, GSocRec models normally use $F$-dimensional latent embedding matrices, i.e., $E_1 \in \mathbb{R}^{N \times D}$ and $E_2 \in \mathbb{R}^{M \times D}$, that can be learned from one-hot encoding of ids (users or items), continuous feature values, discrete feature ids, or their concatenations.

To disentangle social influence and user’s own preferences, GSocRec models further use two different user preference representations, i.e., $E_{ls}$ and $E_{lr}$, to characterize above two types of effects, respectively. For social influence diffusion, GSocRec models leverage the message-passing mechanism of GNN to model the influence diffusion of a user’s friends, denoted as $\mathcal{R}_u = \{v|r_{uv} = 1\}$, on $u$’s preference. Specifically, the $u$’s preference representation after $K$-hop social influence propagation is 

\[
E_{ls}^{(K)}(u) = \text{GNN}\left(E_{ls}^{(K-1)}(u), \left\{E_{lr}^{(K-1)}(v) | \forall v \in \mathcal{R}_u\right\}\right),
\]

where the GNN module can be an arbitrary GNN model such as GCN [23], GAT [44], GraphSAGE [19] and etc. On the other hand, given $u$’s interaction history $\mathcal{P}_u = \{i|p_{ui} = 1\}$, i.e., the set of items that $u$ has interacted before, the preference representation of $u$ after $K$-hop preference propagation is 

\[
E_{lr}^{(K)}(u) = \text{GNN}\left(E_{lr}^{(K-1)}(u), \left\{E_{lr}^{(K-1)}(i) | \forall i \in \mathcal{P}_u\right\}\right),
\]

where another GNN module is used to update $E_{lr}^{(K)}$. The comprehensive representation of user preferences after considering both social influence diffusion and user preference learning is 

\[
E_{l}^{(K)}(u) = \text{Combine}\left(E_{ls}^{(K)}(u), E_{lr}^{(K)}(u)\right),
\]

where $E_{l}^{(0)}(u) = E_{l}(u)$ and the Combine module can be either a mean pooling operation [13, 41] or an attentive aggregation operation [50]. As for item representations, it is also calculated in a similar message-passing manner as 

\[
E_{2}^{(K)}(i) = \text{GNN}\left(E_{2}^{(K-1)}(i), \left\{E_{l}^{(K-1)}(u) | \forall u \in \mathcal{P}_i\right\}\right),
\]

which aggregates information propagated from item $i$’s neighbors $\mathcal{P}_i = \{v|p_{vi} = 1\}$, i.e., the set of users that have interacted with $i$.

Then, based on above $2 \times (K + 1)$ representations of $u$ and $i$, the prediction score of $u$’s preference over $i$, denoted as $\hat{p}_{ui}$, is calculated as
Robust Preference-Guided Denoising for Graph based Social Recommendation

Figure 1: (a) Distribution plot w.r.t. ratio of friends having co-interactions. (b) Our proposed denoising enhanced social recommendation framework.

\[ \hat{p}_{ui} = E_1(u) \cdot E_2(i), \]

where \( E_1(u) = \frac{\sum_{k=1}^{K} E^k(u)}{K+1} \) and \( E_2(i) = \frac{\sum_{k=1}^{K} E^k(i)}{K+1} \).

To train a recommendation model, a widely-adopted approach is to minimize following Bayesian Personalized Ranking (BPR) loss function [34]:

\[ \mathcal{L}^{BPR} = \sum_{(u,i,j) \in \mathcal{P}} -\ln \sigma(\hat{p}_{ui} - \hat{p}_{uj}), \]

where \( \sigma \) is a sigmoid activation function and \( \mathcal{P} \) contains training triples \((u, i, j) \in \mathcal{P}, (u, j) \notin \mathcal{P}\) made up of an observed user-item interaction and another unobserved one. Learning above BPR objective is equivalent to maximizing the likelihood of observing such pairwise ranking relations \( \hat{p}_{ui} > \hat{p}_{uj} \).

2.2 Discussion on Shortcomings

Previous works on GSocRec models generally follow the above paradigm, which focuses on enhancing user preference learning by leveraging the effect of social influence. Therefore, the vital part of GSocRec models is the modeling of social influence diffusion that aims to identify friends with similar interests as the target user and propagate their influence to this user effectively. To achieve this, GSocRec models characterize diversity of social influence by an attention mechanism in message passing, i.e., using GAT in Equation (1) [50, 53, 55]. However, all these methods have to learn a recommendation model on a full \( G_{rs} \), meaning that massive but useless message-passing operations are conducted between those weakly-connected users, which causes a severe efficiency issue. Moreover, since social relations between users are formed because of various reasons, it is quite possible that \( u \) and \( v \) share no interest or have contrast interests in a certain domain. Thus retaining \((u, v)\) can even degrade recommendation accuracy, as it introduces noisy information. In this circumstance, besides the redundancy issue, the noise problem should also be tackled in GSocRec models.

3 PROPOSED METHOD

To resolve the problems brought by redundant and noisy social relations that have hindered effective learning of social recommendation models, we propose the GDMSR method that can serve as a general framework for enhancing an arbitrary GNN based social recommendation model by denoising the original social graph. On the one hand, GDMSR leverages user preference signals to guide the graph denoising process, which helps identify informative social relations in the context of social recommendation. On the other hand, to achieve robust learning on the original social graph with noisy relations, GDMSR further incorporates a self-correcting curriculum learning mechanism that is less impacted by these noisy labeled data. After the above effective and robust graph denoising training process, the obtained model is used to denoise the social network in an adaptive manner.

3.1 Denoising-enhanced Social Recommendation Framework

Instead of directly training a social recommendation model given the whole social network \( G_{rs} \), our proposed GDMSR first learns to denoise \( G_{rs} \) and obtain a much smaller but more informative graph \( G_{rs}^d \). Then a GNN based social recommendation model utilizes \( G_{rs}^d \) as its social graph to enhance user preference learning from the interaction graph \( G_r \).

Denoising user relations for social recommendation. Given an original user-connected graph that contains redundant or noisy user relations (i.e., edges), the denoising task is defined as a link prediction alike task [27] that aims to generate a confidence score \( r_{uv} \) of link formation for each user pair \((u, v)\). Differently, only those observed user relations are checked in the denoising process by ranking according to \( r_{uv} \). Finally, the denoised graph \( G_{rs}^d \) is obtained by removing those \((u, v)\)s with a rather lower \( r_{uv} \).

Training a social recommendation model with the denoised graph. Given \( G_{rs}^d \), as well as \( G_r \), any social recommendation model can be incorporated into GDMSR with no limitation on its structure. Moreover, since two training stages of denoising and recommendation are decoupled, one specific \( G_{rs}^d \) can cooperate with multiple types of recommendation models in a unified way.

Since GDMSR is generally workable for all GNN based social recommendation models discussed in Sec. 2, we focus on introducing the design of graph denoising network in the rest of this section, which has also been illustrated in Figure 2.

3.2 Preference-guided Graph Denoising Network

A key challenge of denoising user-connected social network for recommendation is the lack of groundtruth labels on redundant social relations, and thus every existed link is equally treated when
training the model, making it infeasible to identify informative links from massive data. Current social recommendation models expect to automatically capture the diverse social influence by using a learnable message-passing mechanism (like GAT), which can be ineffective because of indirect supervision from recommendation loss. Since the purpose of denoising is to obtain a sparsified but more informative social graph for user preference learning, we propose to design a preference-guided graph denoising network for solving the social relation redundancy issue in the context of social recommendation.

**GCN-based representation learning.** Similar to GNN based social recommendation models, our designed preference-guided graph denoising network in GDMSR also adopts a GNN based model structure, with the same input of both social graph \( G_s \) and interaction graph \( G_r \). Specifically, the user and item representations after \( K \)-hop propagation, i.e., \( \{ E_1^{(k)} \}_{k=0}^K \) and \( \{ E_2^{(k)} \}_{k=0}^K \) are calculated as in Equation (1) to (4). Since we aim to improve the previous learning approach via attention mechanism, the specific GNN module is GCN instead of GAT.

**Link prediction training.** Given a prediction of the confidence score between two users \( u \) and \( v \), i.e., \( \hat{r}_{uv} \), the link prediction problem aims to minimize the following binary cross-entropy (BCE) alike loss function:

\[
\mathcal{L}_{BCE} = -\sum_{(u,v) \in R} \log(\sigma(\hat{r}_{uv})) - \sum_{(u,v) \notin R} \log(1 - \sigma(\hat{r}_{uv})).
\]

In terms of leveraging user preference as the guiding signal for social graph denoising, GDMSR has the following two designs to better exploit the above useful knowledge.

**Preference-based relation confidence modeling.** A common choice for calculating relation confidence \( \hat{r}_{uv} \) is to design a scoring function \( \phi(\cdot) \), i.e.,

\[
\hat{r}_{uv} = \phi(\{ E_1^{(k)}(u) \}_{k=0}^K, \{ E_1^{(k)}(v) \}_{k=0}^K),
\]

where \( \phi(\cdot) \) can either be a learnable NN (like MLP) or an non-parametric operation (like inner product). However, we argue that this way of relation confidence modeling is not precise, as the inputs \( \{ E_1^{(k)} \}_{k=0}^K \) may take in unrelated information. To find influential friends in the context of social recommendation, it is intuitive to directly compare their interaction history because these characterize the similarity of their preferences. However, above GCN based user representations \( \{ E_1^{(k)} \}_{k=0}^K \) embeds a more comprehensive view of user traits, including user profile (i.e., \( E_1^{(0)} \)), a fusion of user preference and social influence (i.e., \( E_1^{(1)} \)) and higher order ones. To ensure simple but effective guidance from user preferences, our proposed GDMSR only use the interaction history to characterize the relation confidence between two friends. Mathematically, \( \hat{r}_{uv} \) is calculated as follows,

\[
\hat{r}_{uv} = \text{Trf}(\{ E_2^{(0)}(i) \mid i \in \mathcal{P}_u \} \cup \{ E_2^{(0)}(j) \mid j \in \mathcal{P}_v \}),
\]

where a Transformer module [43], denoted as Trf, is used given its power in modeling similarity between two sequences of user interaction history. Specifically, the input is the concatenation of two sequence embeddings (denoted as \( S \)) representing the interaction history of \( u \) and \( v \), respectively, each with a fixed length of \( L \). Since no positional information is required in modeling relation confidence, here we do not apply positional encoding in the Transformer. As for the output, we add a “CLS” alike token at the end of the input sequence and use the corresponding transformer-encoded embedding vector to calculate \( \hat{r}_{uv} \) with a MLP.

**Co-optimization with recommendation loss.** Besides designing the above preference-based relation confidence modeling structure, we further enhance preference guidance by co-optimizing the denoising model with a recommendation loss. As shown in Equation (9), the quality of item embeddings is vital in characterizing relation confidence between users. Motivated by the idea of improving representation quality with self-supervised learning, we propose to add a recommendation loss \( \mathcal{L}_{BPR} \) in Equation (6)) and simultaneously train item embeddings to predict user preferences, which benefits their use in graph denoising in return. Therefore, the final loss function is as follows,

\[
\mathcal{L} = \alpha \mathcal{L}_{BCE} + (1 - \alpha) \mathcal{L}_{BPR},
\]

where \( \alpha \in [0, 1] \) is a hyperparameter that controls the relative importance of link prediction loss and recommendation loss.

In a word, our proposed GDMSR can denoise social relations by exploiting user preference in a two-fold manner, i.e., explicit preference-based relation modeling and implicit co-supervision from preference learning, which both help solve the social relation redundancy issue.

---

**Figure 2: Details of graph denoising process in GDMSR.**
3.3 Robust Denoising of Social Network

Another key challenge of denoising social network for recommendation is the existence of noisy social relations (i.e., friends with no shared preference.) that increases the learning difficulty of the denoising model. For example, suppose there are two connected users \((u, v)\) with distinct preferences, having two sets of interacted items \(\{p_u, p_v\}\) that are less correlated with each other, pushing \(r_{uv} \) in Equation (9) to a large value is generally difficult and may induce a biased model that has to memorize this difficult sample [1]. Therefore, the proposed GDMSR designs a self-correcting curriculum learning mechanism and an adaptive denoising strategy to alleviate this noisy effect and thus achieve robust graph denoising.

Self-correcting curriculum learning mechanism. During the GDMSR learning process, relations with a high probability of being noisy, i.e., with a rather low confidence score predicted by the current denoising model, are removed, and the set of removed relations maintains a dynamic update with the model every several training epochs. This equates following a difficulty-based curriculum that tends to favor the rather easier samples so as to achieve robust learning under noise and keep correcting this curriculum by replacing the more difficult samples with the easier ones according to the current model [64]. Specifically, for a user \(u\), the last \(\eta_u\) of her friends ordered by \(r_{uv}\) are considered as noisy relations to remove from training. This set is updated every \(D\) epochs, where \(D\) is a hyperparameter representing the length of each period in the above self-correcting curriculum.

Adaptive denoising strategy. As illustrated in Figure 2, the trained denoising model is used in the final stage for social network denoising, i.e., predicting the confidence score of each existed social relation and removing those with low confidence. On the one hand, since the training process with noisy labels may be unstable, the robustness of GDMSR is further enhanced by smoothing its prediction results along the training process [5, 22]. Specifically, \(r_{uv}\) at the end of \(k\)th curriculum period, i.e., \(kD\)th epoch, is smoothed by that of the last period. Mathematically, \(r_{uv}\) is updated as follows,

\[
\hat{r}_{uv}(t = kD) = \beta \cdot \hat{r}_{uv}(t = (k-1)D) + (1 - \beta) \cdot \hat{r}_{uv}(t = kD),
\]

where \(\hat{r}_{uv}\) controls smoothness. On the other hand, Motivated by Dunbar’s number theory suggesting an upper limit of a user’s close friends [12], we propose to adaptively denoise the user \(u\)’s social graph based on her friend number, i.e., \(|\mathcal{R}_u|\). Specifically, for each user \(u\), \(\eta_u\) of \(u\)’s friends are removed from the original graph \(G_s\), where we use the same hyperparameter \(\eta_u\) as in the above curriculum design. Mathematically, the denoising ratio \(\eta_u\) is calculated as follows,

\[
\eta_u = \begin{cases} 
0, & \text{if } |\mathcal{R}_u| < \epsilon, \\
\left\lfloor \log_{10}(|\mathcal{R}_u|) \right\rfloor \times \gamma, & \text{else},
\end{cases}
\]

where \(\epsilon\), \(\gamma\) and \(\beta\) are hyperparameters. The core idea behind this formula is that sparsely-connected users (\(|\mathcal{R}_u| < \epsilon\)) can retain all their relations while densely-connected users get to cut more relations in the denoised graph \(G^2_s\), which is more robust comparing uniform dropping regardless of friend number.

In a word, by following the above self-correcting curriculum during the training process and adopting an adaptive manner of denoising, our proposed GDMSR is able to robustly denoise social relations based on user preferences. So far, we have completed the entire denoising process, and the whole algorithm is shown in the Appendix A.

4 EXPERIMENTS

4.1 Experiment Settings

Dataset. We conduct experiments on three open social recommendation datasets, including Ciao, Yelp and Douban. These datasets have been widely used in social recommendation related works [50, 60]. More dataset preprocessing details are in the Appendix B. The statistical information of each dataset is shown in Table 1.

Baseline. Since our GDMSR is able to generate denoised social networks and adapt to arbitrary social recommendation models. For social recommendation model, we choose Diffnet++ [50] and MHNC [60] as baselines. For the denoising method, we compare GDMSR with Rule based approach, NeuralSparse [63] and ESRF [59].

Evaluation. To evaluate the performance of all methods, we use Recall@K and NDCG@K as metrics, where \(K=\{1, 3\}\). According to the suggestions of [62], we use real plus N [3, 35] to calculate the measures. For each user in the test set, we randomly sample 100 items that the user has not interacted with and rank them with the positive samples in the test set.

More implementation details are in the Appendix B.

4.2 Overall Performance

We compare the performance of GDMSR with original social recommendation models without social network denoising and two denoising methods. The results are shown in Table 2. The overall average denoising ratio for all denoising methods ranges from about 5% to 10% depending on the hyperparameters. All results are average value after five repeat experiments. We performed paired t-test between the results of GDMSR and best-performed baseline. All improvements are statistically significant for \(p < 0.01\). It can be observed that GDMSR achieves the best performance in all experiments and the max improvement is up to 10% on one dataset, i.e., Ciao.

Further, we have the following findings. First, compared to using the original social network directly, after denoising the social network through different methods, the performance is improved in all experiments. This shows that in social recommendation, unreliable social relations in social networks are ubiquitous, and not all social relations reflect the homogeneity or social influence of users. Even a simple ruled based denoising method can improve the performance. Second, compared with NeuralSparse which uses both user interaction information and social information for denoising, GDMSR has better performance. This shows that in complicated social networks, it is more effective to use only user preference than to use social relations and user preference for denoising. Finally, the proposed GDMSR can achieve performance improvements on the basis of different models. For example, on the Douban dataset, GDMSR achieved performance improvement of more than 2% compared to the best baseline, using both Diffnet++ and MHNC, which means that GDMSR can identify useless relations without relying...
This proves that our co-optimization module can obtain better item representation directly. The results are shown in Figure 3(a)-(b). It can be found that, on different social recommendation models, as \( \alpha \) increases from 0 to 1, the performance first increases and then decreases. In detail, on the Ciao dataset, when \( \alpha = 0.5 \), the three social recommendation models have achieved their respective best performance. On the Douban dataset, \( \alpha = 0.3 \) or 0.5 works best. This proves that our co-optimization module can obtain better item representation through recommendation loss and further improve the accuracy of the link prediction task. At the same time, when \( \alpha = 0 \), the performance of the model has no obvious advantage compared with the method w/o denoising in Table 2, which shows that it is not enough to only model user preferences, and the denoising model needs the guidance of the link prediction task.

4.3 Ablation Study
Preference-guidance from co-optimization. We propose to jointly optimize the recommendation loss and link prediction loss during denoising training, thereby improving the accuracy of relation confidence modeling. We tested the effect of co-optimization under different weights (\( \alpha \)), and the results are shown in Figure 3(a)(b). It can be found that, on different social recommendation models, as \( \alpha \) increases from 0 to 1, the performance first increases and then decreases. In detail, on the Ciao dataset, when \( \alpha = 0.5 \), the three social recommendation models have achieved their respective best performance. On the Douban dataset, \( \alpha = 0.3 \) or 0.5 works best. This proves that our co-optimization module can obtain better item representation through recommendation loss and further improve the accuracy of the link prediction task. At the same time, when \( \alpha = 0 \), the performance of the model has no obvious advantage compared with the method w/o denoising in Table 2, which shows that it is not enough to only model user preferences, and the denoising model needs the guidance of the link prediction task.

Modeling relation confidence with only preferences. In the relation modeling task, we propose to measure the confidence of the relation between users by only using the user’s feedback, that is, the item representations that the user has interacted with. To verify the effect of this idea, we compare our proposed transformer-based relation confidence modeling module with other three options using different information, including using user representations or pooling the item representation directly. The results are shown in Figure 3(c)-(d). “User-0” and “User-1” indicate using the representation of user nodes in GCN before convolution and from the first layer respectively. We can observe that only using item representation works best, demonstrating the effectiveness of our proposed relation modeling structure, as it ensures simple but effective guidance from user preference (interaction history). At the same time, compared to using user representation, only using item representation for pooling can achieve similar results. This again shows that the denoising model cannot effectively model the users’ social relation confidence when integrating the user-side information including their profile (i.e., \( E_{u,1}^{(0)} \)) and social influence (i.e., \( E_{u,1}^{(1)} \)).

Denoising robustness. In robust denoising module, we propose an adaptive denoising strategy, which makes the denoising ratio of users with more friends larger. To verify the effect of this strategy, we compare strategies that use the same denoising ratio (\( \eta_u \)) for all users. As shown in Figure 4 and Appendix B.8, the performance w.r.t. Recall@1 in different cases consistently shows a trend of first increasing and then decreasing, indicating that proper denoising of social network can improve effectiveness of social influence, thereby improving the recommendation performance. Meanwhile, compared with non-adaptive denoising, the proposed GDMSR significantly outperforms in nearly all cases with different denoising ratio. More importantly, our proposed GDMSR is able to achieve further sparsification without hurting original recommendation accuracy, which produce a memory-efficient social graph that is vital in practical applications. Specifically, GDMSR combined with Diffnet++ is able to achieve 20-40% of relation reduction while maintaining an accuracy improvement. Comparatively, this reduction value drops to about 10% when combined with MHCN, which is still of large practical value in commercial recommender systems. Since

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#Users</th>
<th>#Items</th>
<th>#Interactions</th>
<th>#Relations</th>
<th>Interaction Density</th>
<th>Relation Density</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ciao</td>
<td>7,355</td>
<td>17,867</td>
<td>140,628</td>
<td>111,679</td>
<td>0.11%</td>
<td>0.21%</td>
</tr>
<tr>
<td>Yelp</td>
<td>32,827</td>
<td>59,972</td>
<td>598,121</td>
<td>964,510</td>
<td>0.03%</td>
<td>0.09%</td>
</tr>
<tr>
<td>Douban</td>
<td>2,669</td>
<td>15,940</td>
<td>535,210</td>
<td>32,705</td>
<td>1.26%</td>
<td>0.46%</td>
</tr>
</tbody>
</table>

Table 1: Basic information of datasets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Basemodel</th>
<th>Method</th>
<th>R@1</th>
<th>R@3</th>
<th>N@3</th>
<th>R@1</th>
<th>R@3</th>
<th>N@3</th>
<th>R@1</th>
<th>R@3</th>
<th>N@3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ciao</td>
<td>LightGCN</td>
<td>w/o denoising</td>
<td>0.2298</td>
<td>0.0785</td>
<td>0.2071</td>
<td>0.5861</td>
<td>0.2774</td>
<td>0.5804</td>
<td>0.4321</td>
<td>0.1696</td>
<td>0.4156</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rule based</td>
<td>0.2742</td>
<td>0.1109</td>
<td>0.2639</td>
<td>0.6031</td>
<td>0.3072</td>
<td>0.5897</td>
<td>0.5165</td>
<td>0.2156</td>
<td>0.4988</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NeuralSparse</td>
<td>0.2806</td>
<td>0.1123</td>
<td>0.2677</td>
<td>0.6230</td>
<td>0.3228</td>
<td>0.5996</td>
<td>0.5358</td>
<td>0.2489</td>
<td>0.5172</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ESRF</td>
<td>0.2864</td>
<td>0.1153</td>
<td>0.2734</td>
<td>0.6383</td>
<td>0.3289</td>
<td>0.6054</td>
<td>0.5470</td>
<td>0.2226</td>
<td>0.5102</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GDMSR</td>
<td>0.3020</td>
<td>0.1244</td>
<td>0.2821</td>
<td>0.6449</td>
<td>0.3291</td>
<td>0.6102</td>
<td>0.5614</td>
<td>0.2540</td>
<td>0.5297</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( \Delta )</td>
<td>5.26%</td>
<td>3.93%</td>
<td>3.11%</td>
<td>1.03%</td>
<td>0.06%</td>
<td>0.79%</td>
<td>2.63%</td>
<td>2.05%</td>
<td>1.98%</td>
</tr>
<tr>
<td>Yelp</td>
<td>LightGCN</td>
<td>w/o denoising</td>
<td>0.2300</td>
<td>0.0884</td>
<td>0.2297</td>
<td>0.6991</td>
<td>0.3252</td>
<td>0.6364</td>
<td>0.6198</td>
<td>0.3167</td>
<td>0.5933</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Rule based</td>
<td>0.2301</td>
<td>0.0916</td>
<td>0.2311</td>
<td>0.6966</td>
<td>0.3234</td>
<td>0.6347</td>
<td>0.6082</td>
<td>0.3372</td>
<td>0.5980</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NeuralSparse</td>
<td>0.2461</td>
<td>0.1034</td>
<td>0.2540</td>
<td>0.7012</td>
<td>0.3288</td>
<td>0.6352</td>
<td>0.6206</td>
<td>0.3349</td>
<td>0.6011</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ESRF</td>
<td>0.2495</td>
<td>0.1028</td>
<td>0.2568</td>
<td>0.6927</td>
<td>0.3298</td>
<td>0.6344</td>
<td>0.6194</td>
<td>0.3244</td>
<td>0.5995</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GDMSR</td>
<td>0.2618</td>
<td>0.1138</td>
<td>0.2632</td>
<td>0.7036</td>
<td>0.3405</td>
<td>0.6434</td>
<td>0.6396</td>
<td>0.3496</td>
<td>0.6137</td>
</tr>
<tr>
<td></td>
<td></td>
<td>( \Delta )</td>
<td>4.93%</td>
<td>10.06%</td>
<td>2.50%</td>
<td>0.34%</td>
<td>3.24%</td>
<td>1.10%</td>
<td>3.06%</td>
<td>2.10%</td>
<td>3.68%</td>
</tr>
</tbody>
</table>

Table 2: Overall performance of our proposed method on different recommendation methods.
In addition, we also compare the effect of choosing different values of co-optimization weight and inference efficiency.

Figure 3: (a)-(b) Performance on different $\alpha$ for co-optimization. (c)-(d) Performance comparison among different relation confidence modeling structures.

Figure 4: Analysis of denoising robustness with respect to recommendation accuracy under different denoising ratio.

Figure 5: (a) Retaining ratio of social relations adopting different denoising strategies (synthetic). (b) Efficiency comparison w.r.t. social graph related inference time (Yelp).

MHCN relies on mining social motifs and leverages hypergraph convolution model to characterize their effects, graph denoising may bring larger impacts on MHCN because of structural changes. In addition, we also compare the effect of choosing different value of hyperparameter $y$. Due to space limit, please refer to Appendix B.8 for details.

4.4 Denoising Capability on Synthetic Data

To further evaluate denoising capability of our proposed GDMSR, we construct a synthetic dataset (based on Ciao) with both observed social relations and fake social relations (i.e., we randomly set some unconnected user pairs to be connected) and evaluate whether GDMSR is able to learn a powerful graph denoising model that can effectively discriminate a given relation label. Specifically, we inject the fake social relations with the same number of those observed ones, and further combine these corrupted relation data with original interaction data to train GDMSR. Then we record the retaining ratio of each part of relation data during the denoising process, and present the results in Figure 5(a). Here we also compare with two variants, i.e., without adaptive denoising (shorten as “w/o ad”) and without adaptive denoising and self-correcting curriculum learning (shorten as “w/o ad sc”). The results clearly indicate that GDMSR, even though under explicit noise of corrupted relation labels, can still maintain learning robustness by removing more fake relations (16%) than observed relations (9%). In other words, since we inject rather strong noise (50% of relation labels are corrupted), GDMSR manages to achieve an moderately good accuracy of 64% (16/16+9), while other two options cannot produce a workable denoising module under this level of label noise.

4.5 Practicality Evaluation

Computation efficiency. Our proposed GDMSR can produce a much sparser social graph (e.g., 20–40% smaller on Yelp), which can improve computation efficiency in inference period. In Figure 5(b) we present an inference time analysis of social graph related computations (social-network-side GNN), which is conducted on Yelp dataset by inferring 50w records. The results clearly demonstrate the increasing improvement of computation efficiency as denoising ratio goes larger.
Zero-shot denoising without frequent training. In industry practice, it is rather time consuming and nearly infeasible to frequently update graph denoising model given the massive social relation data (up to hundreds of millions in Facebook and Wechat). Therefore, GDMSR is expected to support denoising training in a less frequent manner, i.e., produce high-quality distilled social graph without training on newly coming interaction data [28]. To evaluate this practicality, we design another experiment by only using 30% of interaction data to train denoising module of GDMSR and then use all interaction data as input to denoise social graph. This setting simulates the fact that GDMSR cannot be timely updated in practice. Instead, we may use the old GDMSR model but the new interaction data to directly denoise current social graph in a zero-shot-like manner. Then, the denoised graph is used to train the social recommendation models and results are listed in Table 3. It can be observed that GDMSR is still able to effectively denoise the social network even when only 30% of the training samples are used. In most cases, the recommendation accuracy can be improved compared to without denoising, with the highest improvement reaching 23%, demonstrating that GDMSR has high scalability in terms of industrial practice. In fact, this is mainly owing to our design of preference-guide denoising with only interactions, which is more robust and less influenced by other contextual changes.

5 RELATED WORK

5.1 Graph Denoising for Social Recommendation

Early works use simple statistics like number of co-interactions [31] to indicate the confidence degree of social relations, which is generally ineffective due to data sparsity [40]. In order to capture observed diversity of social influence, attention mechanism [50, 55], fine-grained contextual information [14], expectation-maximization method [47] or reinforcement learning [11] can be leveraged to learn adaptive weights among various friends. Although above techniques can be applied in prevalent GSocRec models, they generally fall short of learning the degree of social influence effectively due to a lack of groundtruth labels. From a view of general graph machine learning, graph sparsification methods [7, 63] learn to drop edges from a general graph so as to achieve robust learning, which does not consider the specific social recommendation context. Moreover, graph structure learning methods [29, 30, 61] are optimized to not only drop but also add edges, which are too time-consuming for social recommendation as they generally operate on a huge space with cartesian product between users. As a specific extension in social context, there are a series of works that propose to not only remove observed social relations with low quality but also add a few potential relations [24, 56, 57, 59]. This idea has also been adopted in dynamic recommendation scenarios like session recommendation by starting with observed social relations and updating relations during training process [38, 49]. In addition, the effectiveness of social recommendation can also be improved by using contextual features [26] or multi-view learning [25]. Different from above works, our proposed GDMSR method can produce a distilled social graph, not only sparser but also more informative, that is more suitable for real-world social recommender systems. In experiments we also compare with one competitive method above [59] to demonstrate effectiveness of GDMSR.

5.2 Denoising for Recommendation

In general recommender systems, another trending topic is feedback denoising that enables more robust preference learning [21, 42]. Since there exists heterogeneity in users’ manner of expressing preferences, it is non-optimal to set a unitary threshold for either explicit feedback or implicit feedback. For example, users differ in their criterion of rating items with low scores [45] or not clicking [8–10]. There are also some works to solve this problem through causal inference [16]. To deal with this problem of learning from noisy labels [37], existing works adopt techniques like adding graph-based priors [39], cross-model agreements [48] and graph sparsification [17] to denoise observed user-item interaction data. Different from above works, we focus on denoising social network, i.e., user-user relation data, so as to leverage power of social homophily and social influence in a more effective and efficient way.

6 CONCLUSION AND FUTURE WORK

In this paper, we approach the problem of graph based social recommendation by rethinking the reliability of the social network for learning GNN recommendation models. Since there is no reliably labeled data, we choose to utilize user-item feedback that reflects user preference to filter noisy social relations, which, in turn, improves the preference learning. Experiments on three public benchmark datasets and different social recommendation models not only show the effectiveness of the denoising method but also demonstrate our proposed solution can serve as a general framework. Precisely, the improvement is steady and significant, up to 10%.

As for the future plan, one interesting direction is to deeply analyze the difference between the original social network and the denoised one, supported by the social network theory [4].
ACKNOWLEDGMENTS

This work was supported in part by the National Key Research and Development Program of China under 2020AAA0106000, and the National Natural Science Foundation of China under 61971267, U1936217, 61972223, 62272262.

REFERENCES


A TRAINING PROCESS OF GDMSR

The overall process of GDMSR is shown in Algorithm 1.

Algorithm 1: Training process of GDMSR

Require: social relations set \( R \), interactions set \( P \)
1: for epoch = 1 to MAX_EPOCH do
2:   Generate training batch \( B \) from \( P \)
3:   for \((u, i) \in B\) do
4:     sample \((u, j), (u, v)\) and \((u, w)\) with \( j \notin P_u, v \in R_u \) and \( w \notin R_u \)
5:     Calculate recommendation loss \( L^{BPR} \) by Eq.6
6:     Calculate link prediction loss \( L^{BCE} \) by Eq.7
7:     Calculate final loss \( L \) and update parameter by Eq.9
8:   end for
9:   if epoch\%D == 0 then
10:      for \((u, v) \in R\) do
11:         Calculate \( r_{uv}(t = kD) \) by Eq.8 and Eq.10
12:      end for
13:   end if
14:   for \( u \in U \) do
15:      Calculate \( \eta_u \) by Eq.11
16:      Remove last \( \eta_u \) of \( u \)'s friends ordered by \( r_{uv}(t = kD) \)
17:   end for
18: end for

B EXPERIMENT DETAILS

B.1 Dataset preprocessing

For all datasets, we only retain the samples with a score of 4 or 5 as positive samples. Besides, we filter users and items according to the number of interactions and the number of users’ friends and only retain samples related to users and items with high activity. For the Yelp dataset, we retain the user attributes, and since the original data does not contain user attributes for the other two datasets, we only use the user id as a feature.

B.2 Baselines

Here is an introduction to the baseline methods we used:

Diffnet++. This model uses graph attention networks to model diverse social influences among various friends.

MHCN. This model uses hypergraph convolution methods to capture different impacts of social motifs on preference learning among friends.

Rule based approach. Based on the social homogeneity assumption, the similarity between users can be measured according to the items that the users interact with. Therefore, we remove those relations with a few items that users have both interacted with.

NeuralSparse. This is an end-to-end adaptive graph sparsification method that learns a differentiable sampler from empirical data by using Gumbel softmax.

ESRF. This end-to-end social recommendation framework leverages Generative Adversarial Nets (GAN) so as to generate more densely-connected social graphs for social recommendation.

B.3 Running Environment

The experiments are conducted on a single Linux server with AMD Ryzen Threadripper 2990WX@3.0GHz, 128G RAM and 3 NVIDIA GeForce RTX 2080TI-11GB. Our proposed GDMSR is implemented in PyTorch 1.10 and Python 3.7. Social recommendation models and baselines including SocialGCN, Diffnet++, MHCN and NeuralSparse are implemented in Tensorflow 1.14 and Python 3.7.

B.4 Assets

The code of our proposed GDMSR is included in the supplementary materials. Diffnet++, MHCN and NeuralSparse are implemented based on open source code. It should be noted that NeuralSparse is not an official implementation. The implementation of SocialGCN is based on Diffnet++. We removed the attention module and made some corresponding modifications according to the original paper [52].

B.5 Implementation Detail.

For all experiments, the embedding size and batch size are set to 8 and 1024 respectively, and they all use the Adam optimizer. All social recommendation models use BPR loss. Hyper-parameters like learning rate and dropout are obtained by grid search. The search range is as follows: the learning rate is \{0.005, 0.001, 0.0005, 0.0001\}, the dropout is \([0, 1]\), and the weights \((\alpha)\) for the co-optimization in GDMSR are \{0.1, 0.3, 0.5, 0.7, 0.9\}. For adaptive denoising strategy, the threshold \((\epsilon)\) is set to 5, the search range of \( \gamma \) and \( R \) are \{0.5, 1.0, 2.0\} and \{0.01, 0.015, 0.02, 0.025, 0.03\} respectively. For NeuralSparse, the number of neighbors sampled by each node is searched from \{30, 50, 100\}. For all experiments of GDMSR, we train the denoising model for 200 epochs, at which time \( L^{BCE} \) was basically stable. For all datasets, the interaction history of each user has been preprocessed to have a fixed length, i.e., padding for those shorter than \( L \) and truncation for excess ones. The truncation follows a descending order based on item popularity in datasets.

B.6 Loss Curve of Denoising Training

In denoising training, for all datasets, we train for 200 epochs so that \( L^{BCE} \) is stable, and the loss curves are shown in the Figure 6. We can observe that after 200 epochs of training, the loss function has basically stabilized, indicating that the denoising training has been completed.

B.7 Experiment details

In the experiments, we control the number of social relations that are finally removed by controlling different hyper-parameters for GDMSR and baselines, respectively. Specifically, for GDMSR, we control the \( \epsilon, \gamma \) and \( R \). For Rule based approach, we control the ratio of friends removed per user. For NeuralSparse, We control the hyperparameter \( K \), which is the maximum number of neighbors that each node can receive message. For ESRF, We also control the number of neighbors each node uses, in the source code, this value is set to 100, and no hyperparameters are provided to tune. We made modifications directly in the source code.

---

1[https://github.com/PeiJieSun/diffnet/tree/master/Diffnet%2B%2B]
2[https://github.com/Coder-Yu/RecQ]
3[https://github.com/flyingdoog/PTDNet/tree/main/NeuralSparse]
To verify the robustness of the adaptive denoising strategy, we compared the strategy to the method that assigns the same denoising ratio for all users, and we study the effect of the different $\gamma$ on the performance. The results are shown in Figure 7(b)-(d). When $\gamma = 0.5$, the overall performance is the best. This means that except for those users with very few friends, most users are assigned a larger denoising ratio to achieve better performance.