A Scalable Social Recommendation Framework with Decoupled Graph Neural Network

Ke Tu[1], Zhengwei Wu[1], Binbin Hu[1], Zhiqiang Zhang[1], Peng Cui[2], Xiaolong Li[3], and Jun Zhou[1]

[1] Ant Group
{tuke.tk,zejun.wzw,bin.hbb,lingyao.zzq,jun.zhoujun}@antgroup.com
[2] Tsinghua University cuip@tsinghua.edu.cn
[3] Alibaba Group xl.li@alibaba-inc.com

Abstract. Social relationships are usually used to improve recommendation quality, especially when users' behavior is very sparse in recommender systems. Most existing social recommendation methods apply Graph Neural Networks (GNN) to capture users' social structure information and user-item interaction information. However, the GNNs need to conduct expensive neighborhood propagation, leading to scalability issues. Some recent works pointed out that the GNNs can be simplified via decoupling. Therefore, we propose a scalable framework for social recommendation to decouple the model into two stages, Gumbel-based feature propagation and self-supervised multi-representation fusion. In the first stage, since the similarity between friends will not change frequently, we pre-train a Gumbel sampling-based attention model offline to learn the importance of each social relation and use the importance as the weight to aggregate the feature during pre-computation. Due to the diversity of user interests, the features are propagated upon different propagation layers to capture information with different aspects. In the second stage, we use the aggregated representations as inputs and fuse the inputs by an attention mechanism to obtain comprehensive embeddings on the online mode to update daily. Besides, we use a contrastive learning way to enrich users' information. Moreover, extensive experimental results demonstrate the scalability and effectiveness of our framework over state-of-the-art algorithms.

Keywords: Decoupled GNN · Social Recommendation · Scalability

1 Introduction

Nowadays, recommender systems are widely used in the industry, such as e-commerce, advertising, and search engines. Due to the widespread existence of data sparsity and cold start issues, recent studies seek to introduce social networks into recommender systems, known as *Social Recommendation*, for enriching "cold" users' information via their neighbors, guided by the principle of homophily [10] that friends tend to share similar interests. Meanwhile, *Graph Neural Networks* (GNNs) [4] has become newly state-of-the-art for modeling

network structure, and its effectiveness in facilitating social recommendation has been well studied [19]. Roughly speaking, existing works mainly extend prevailing GNN architectures to adapt for various social recommendation scenarios through social diffusion [15], adversarial training based denoising [19, 17] and self-supervised learning [16, 6]. Notwithstanding the promising performance in benchmarks, we are crucial about following fundamental questions, aiming at the industrial settings of social recommender systems based on GNNs.

- *Is the recently emerging GNN-based social recommendation approaches suitable for real-world industrial applications?* Current GNN based social recommender systems, such as Diffnet++ [15] and MHCN [20], follow the typical GNN architecture, which recursively obtains user/item representations via nonlinear transformation of its neighbors. Intuitively, the heavy design causes both high computational cost and large space requirement, which severely threatens the scalability of the industrial recommender system. Moreover, the scale of recommender systems, as well as the social networks, is enormous in real-world business and is expected to be exploited for training and inference in a tolerable delay. Such a dilemma highlights an inevitable bottleneck to incorporating GNNs into social recommendation in practice.
- *Is social/interaction relations in reality reliable enough to unfold the strength of GNNs?* Unfortunately, real-world relational data is not only noisy (i.e. users may misclick some unwanted items) but also unreliable (i.e. the strength of social ties is uncertain) in most cases. Due to the recursive embedding propagation of GNNs, the capability of GNN-based social recommendation clearly deteriorates by passing unsatisfying or even harmful information.
- *How to address the problem of the information asymmetry among different views in the social recommendation?* By incorporating social relations, users may naturally have multiple views of neighbors such as friends and clicked items. Due to the sparsity issue, the user may have very few neighbors in some views. The final representations may be mainly influenced by the rich information view by the uniform neighbor aggregation of GNNs. It is important to enrich the sparse context for balancing different views to solve the information asymmetry issue.

In this paper, we borrow the basic idea of GNN decoupling and propose a **S**calable **S**ocial **R**ecommendation framework named **SSR** for the online recommendation system. We decouple our model into two-stage, Gumbel-based [7] feature propagation and self-supervised multi-representation fusion. In the first stage, we pre-process the feature propagation offline. To deal with the noisy and unreliable social relations, we pre-train a Gumbel-attention based GNN to choose the most critical social neighbors. Then we propagate features among the selected neighbors upon different propagation layers to keep the diversity of node representations. In the second stage, we use the pre-processed propagated features as input and apply an attentive aggregation to summarize representations from different propagation layers for users and items on the online daily updated datasets. In terms of the information asymmetry issue among different views, we come up with the contrastive learning [5] module to learn their representations.

As the feature propagation in the first stage is pre-processed only once offline, and the online deep neural network in the second stage is efficient, our model can be scaled to large network datasets.

It is worthwhile to highlight the following contributions of this paper:

- We highlight the importance of the scalability problem in the social recommendation and propose a novel model named **S**calable **S**ocial **R**ecommendation framework (**SSR**) for the recommendation system which is scalable for massive industry scenarios. To the best of our knowledge, our model is the first work to introduce decoupled GNN into social recommendation.
- We decouple our model into the offline and online stage. In the offline pretrain stage, we use Gumbel attention to obtain the most useful social friends for solving unreliability issue of social relations. And in the online daily updated stage, we design a model to merge representations from different path-guided views and enrich the interest information for all users by contrastive learning to deal with information asymmetry issue.
- Extensive experimental results, including three common-used public datasets and a large-scale industry dataset, demonstrate the scalability and effectiveness of our framework over the state-of-the-art algorithms.

2 RELATED WORK

2.1 Graph Neural Network

The Graph Neural Networks(GNNs) [22] has been proposed to deal with network data in an end-to-end manner. [1] introduces the graph signal process and defines the convolution in the Fourier spectral domain. GAT [14] applies a self-attention strategy to aggregate nodes in neighborhoods with different weights. Benefit from the strong network representation ability, GNNs are widely used in industry scenarios. Despite their effectiveness, the graph neural networks used in previous works lead to scalability issues in real industry massive scenarios. It is worth mentioning that recently deep insights into GNNs have shed some light on the design of scalable GNNs in practice, which performs simplification via decoupling [13, 21]. They decouple GNNs into two-stage, feature propagation, and non-linear mapping. For example, SIGN [13] proposes to delete the non-linear operators and simplify GNNs into an MLP operating on pre-computed concatenated multi-hop averaged features. Moreover, NARS [21] extends them into a heterogeneous version. However, its effectiveness has not been explored for social recommendation. Most notably, the unreliability of relations could lead to terrible degradation in recommendation performance since neighbor averaging-based strategies are commonly adopted for decoupling.

2.2 Social Recommendation

Due to the social network principle of homophily, users' interests are often influenced by their friends. Recent works aim to introduce social networks into the

recommender system to alleviate the issue of data sparsity. The early social recommendation models can be categorized into two groups, factorization methods, and regularization methods. The factorization-based methods, such as SoRec [8], factorize the user-item rating matrices and user-user social matrices and map the factorized user and item representations into the same latent space. As for the regularization methods [9], they add a social regularization loss term, which guides the users' representations are closed to the representations of their friends, to the user preference ranking loss term. Based on these ideas, recent works introduce graph neural networks to describe the social network and preference network. Diffnet++ [15] model the recursive dynamic social diffusion to apply information from social neighbors. ESRF [19] and RSGAN [17] use adversarial training to generate reliable friends for denoising social relations. S^2-MHCN [20] apply hypergraph to capture motif structures to solve the multi-faceted social relations. However, in the real world daily updated recommender systems, none of them can well solve the scalability, unreliability, and information asymmetry issues. In light of the above considerations, we develop a GNN-based social recommender system by graph decoupling to solve all the previous issues.

3 THE PROPOSED FRAMEWORK

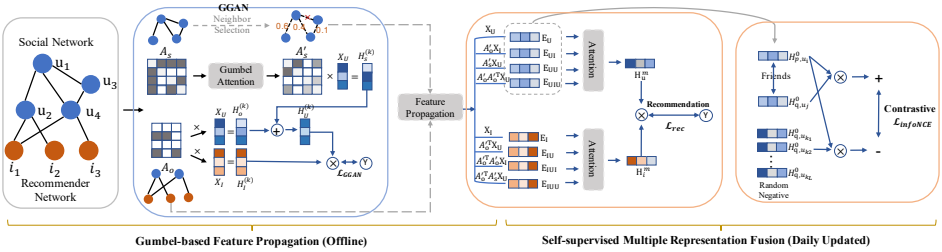

Fig. 1. The Framework of Our Proposed Model SSR.

In this section, we will introduce our proposed framework named **S**calable **S**ocial **R**ecommendations Framework (**SSR**) shown in Figure 1. Our framework consists of two parts, Gumbel-based feature propagation and self-supervised multi-representation fusion.

3.1 Notations

Let $U = \{u_1, u_2, ..., u_m\}$ and $I = \{i_1, i_2, ..., i_n\}$ denotes the sets of users and items in a traditional recommendation scenario, where m is the number of users and n is the number of items. We define the user feature matrix and item feature matrix as $\mathbf{X_U} \in R^{m \times d}$ and $\mathbf{X_I} \in R^{n \times d}$. In social recommendation, there are two

networks, user-item interaction network $G_o = \{V_o = \{U, I\}, \mathbf{A}_o\}$ and social network $G_s = \{V_s = U, \mathbf{A}_s\}$ where V_o, V_s denote the node sets and $\mathbf{A}_o$, $\mathbf{A}_s$ denote the adjacency matrix. If user u has interaction with item i, such as clicked and purchased, $\mathbf{A}_{o,ui}$ equals 1 otherwise 0. $\mathbf{A}_{s,u_1u_2} = 1$ denotes user u_1 and u_2 have social relations. $\mathcal{N}_o(u)$ and $\mathcal{N}_s(u)$ are user u's neighbors in network G_o and G_s, respectively. $\mathbf{Y} \in R^{m \times n}$ is the feedback matrix and y_{ui} which user u's feedback on the item i is either 1 (positive) or 0 (negative or unknown). In our setting, there may be two user-item interaction networks G_o and G_o^m. G_o is historical behavior network for first stage pre-training. And G_o^m is the recent daily updated user-item interaction network. We give the definition of the task Social Recommendation as follows:

Definition 1 (Social Recommendation). *Given the social network $G_s = \{V_s = U, \mathbf{A}_s\}$, user-item interaction network $G_o = \{V_o = \{U, I\}, \mathbf{A}_o\}$ and the features of users $\mathbf{X}_U$ and items $\mathbf{X}_I$, the social recommendation aims to learn a predictive model that effectively forecasts the future user-item interaction.*

3.2 Graph Model Decoupling

Most existing graph neural networks are composed of feature propagation and non-linear mapping. In the model, a non-linear mapping follows a feature propagation step as follows:

$$\mathbf{T}^{(k)} = \mathbf{A}\mathbf{H}^{(k)}, \tag{1}$$

$$\mathbf{H}^{(k+1)} = \sigma_k(\mathbf{T}^{(k)}\mathbf{W}^{(k)}), \tag{2}$$

where $\mathbf{A}$ is sparse adjacency matrix, $\{\mathbf{H}^{(k)}, k = 1, 2, ..., K\}$ is the hidden representations and we set $\mathbf{H}^{(0)}$ equals feature matrix $\mathbf{X}$ and $\mathbf{W}^{(k)}$ is the weight matrix in layer k. The σ_k is a non-linear function. For different graph neural networks, the weighted adjacency matrix $\mathbf{A}$ may be different. For example, GCN [4] and GAT [14] re-weight the adjacency matrix by degree and attention matrix respectively. Since there is a massive sparse-dense matrix multiplication in Equation 1, it is very time-consuming. As LightGCN [3] pointed out, the primary role of GNNs is based on the message-passing layer instead of the non-linear mapping layer. After deleting the non-linear mapping between layers, the GNNs can be simplified as: $\mathbf{H}^{(K)}_{LightGCN} = \sigma((\alpha_1\mathbf{A}\mathbf{X} + \alpha_2\mathbf{A}^2\mathbf{X} + ... + \alpha_K\mathbf{A}^K\mathbf{X})\mathbf{W})$. In such a situation, the GNNs can be decoupled as feature propagation and non-linear mapping. In the feature propagation step, we can pre-compute all the feature propagations such as $\mathbf{A}\mathbf{X}, \mathbf{A}^2\mathbf{X}, ..., \mathbf{A}^K\mathbf{X}$ since there are no trainable variables. This process runs only once. In the non-linear mapping step, we can merge the propagated features and apply a deep neural network to learn weights $\mathbf{W}$. It can be efficient and scalable. To introduce decoupled GNNs into social recommendation, we enrich these two steps into Gumbel-based feature propagation and self-supervised multi-representation fusion. The first step of our model is pre-trained offline and runs only once to obtain stable social relationships. And the second step updates daily online to use the stable social relationships and the new arrival datasets to forecast the future user-item interactions.

3.3 Gumbel-based Feature Propagation

Graph Gumbel Attention Network Different from the traditional recommendation problems, there are social relations in our setting. In real-world social networks, the relations between users may be noisy and unreliable. However, the sparse weighted adjacency matrix $\mathbf{A}$ used in previous works is usually constant and human-designed. To find the useful neighbors who contribute most to the task, we need sparse attention to sparsify the social networks to obtain a sparse and most useful weighted adjacency matrix $\mathbf{A}'$. We propose a pre-trained model named Graph Gumbel Attention Network (GGAN) to obtain the learned adjacency matrix $\mathbf{A}'$. Since the social relations between users are stable in the middle period and users' interaction behaviors with items change daily, the goal of the pre-train model is to obtain the denoised and sparse social adjacency matrix. To make the discrete neighbor selection differentiable, we use Gumbel sampling [7] along with the reparametrization trick to produce a relaxation of the one-hot vector to represent the selected new neighbor. For a user u, we first calculate the edge attentions of each neighbor as follows:

$$\alpha_{uv} = \text{softmax}_{v \in \mathcal{N}_s(u)}(\text{LeakyReLU}(\mathbf{W}_a^T[\mathbf{X}_u \| \mathbf{X}_v])), \tag{3}$$

where $\mathbf{W}_a$ is trainable weight matrix, $\|$ is the concatenate function and LeakyReLU is a non-linear activation. We denote the user u's attention vector as $\boldsymbol{\alpha}_u = [\alpha_{u1} \| \alpha_{u2} \| ... \| \alpha_{um}]$. Then the Gumbel-sampled relaxation of one-hot vector to select the important neighbor is as follows:

$$\mathbf{v}_u = \frac{\exp\left(\left(\log \boldsymbol{\alpha}_u + \mathbf{g}\right)/\tau\right)}{\sum_{v \in \mathcal{N}(u)} \exp\left(\left(\log \alpha_{uv} + \mathbf{g}_v\right)/\tau\right)}, \tag{4}$$

where $\mathbf{g}$ follows the Gumbel(0, 1) distribution[4]. $\tau \in (0, +\infty)$ is the temperature hyper-parameter and it controls the sparsity of the one-hot vector. When τ is smaller, $\mathbf{v}_u$ is closer to a one-hot vector. We conduct Gumbel sampling for T times in Equation 4 to obtain the Gumbel attention-based adjacency matrix:

$$\mathbf{A}_s^{'} = \mathbf{A}_s \odot \sum_{i=0}^{T-1} \mathbf{V}^{(i)}/T, \tag{5}$$

where $\odot$ is the dot product, $\mathbf{V}$ is a matrix of all Gumbel attentions, and $\mathbf{A}_s$ is the sparse adjacency matrix of social network G_s. After that, we aggregate social neighbors' information like Equation 1 and 2 to obtain social-based hidden representation $\mathbf{H}_s^{(k+1)}$. For the user-item network, we use the origin sparse adjacency matrix $\mathbf{A}_o^{'} = \mathbf{A}_o$ to aggregate graph information to obtain interest-based hidden representation $\mathbf{H}_o^{(k+1)}$ in the same way. We combine the final $\mathbf{H}_s^{(K)}$ and $\mathbf{H}_o^{(K)}$ followed an MLP layer to obtain the final user representation $\mathbf{H}_U^{(K)}$. Then

[4] Gumbel(0, 1) can be transformed by uniform distribution, $g = -log(-log(u)), u \in \text{Uniform}(0, 1)$.

we use the inner product of the final user and item representation to predict the feedback labels:

$$\mathcal{L}_{GGAN} = \sum_{(u,i)\in\mathcal{O}} \|\mathbf{H}_u^{(K)}\mathbf{H}_i^{(K)T} - \mathbf{Y}_{ui}\|^2, \tag{6}$$

where $\mathcal{O}$ is the set of all samples including positive samples and randomly sampled negative samples. After optimizing GGAN by Adam, we can obtain the learned sparse social adjacency matrix $\mathbf{A}_s^{'}$.

3.4 Social Representations with Multiple Path-guided Views

To better keep the diversity of node representations, we use multiple parameter-free propagation layers to generate multiple representations as the input of the second stage. To keep users' short real-time interests, we use the recent daily updated user-item interaction network $\mathbf{A}_o^{'}$ along with the previous learned social weighted adjacency matrix $\mathbf{A}_s^{'}$ to aggregate neighbors. We aggregate the feature in the way of propagation path sets $M_U = \{U, UI, UU, UIU\}$ and $M_I = \{I, IU, IUI, IUU\}$ as follows:

$$\begin{gathered}\mathbf{E}_U = \mathbf{X}_U; \mathbf{E}_I = \mathbf{X}_I; \mathbf{E}_{UI} = \mathbf{A}_o^{'}\mathbf{E}_I; \mathbf{E}_{IU} = \mathbf{A}_o^{'T}\mathbf{E}_U; \\ \mathbf{E}_{UU} = \mathbf{A}_u^{'}\mathbf{E}_U; \mathbf{E}_{IUU} = \mathbf{A}_o^{'}\mathbf{E}_{UU}; \mathbf{E}_{UIU} = \mathbf{A}_o^{'}\mathbf{E}_{IU}; \mathbf{E}_{IUI} = \mathbf{A}_o^{'T}\mathbf{E}_{UI}.\end{gathered} \tag{7}$$

The propagation paths can be categorized into four overlapped groups according to the implicit semantics. U and I are the origin feature of users and items. UI and IU contains users' behavior information. UIU and IUI apply high-order network information. UU and IUU rich users' information by social relations. Since the propagation process is parameter-free, this generation can be pre-processed before training.

3.5 Self-supervised Multi-Representation Fusion

In the second stage, we fuse the pre-processed features $\{\mathbf{E}_p, p \in M_U \cup M_I\}$ and capture the users' interests in an end-to-end manner. These representations cover different aspects of users and items. We use an attention mechanism to obtain the cross information. The attention aims to learn the importance of each propagated representation of p. For simplicity, here we only present the attentive aggregator for users, and the same aggregator will be used for items:

$$\begin{gathered}\mathbf{F}_p^0 = \text{LeakyReLU}(\mathbf{E}_p\mathbf{W}_p^0 + \mathbf{b}_p^0), \hat{\beta}_p = \text{LeakyReLU}(\mathbf{F}_p^0\mathbf{W}_p^a + \mathbf{b}_p^a), \\ \beta_p = \frac{\exp(\hat{\beta}_p)}{\sum_{p\in M_U}\exp(\hat{\beta}_p)}, \mathbf{F}_U^m = \sum_{p\in M_U} \beta_p * \mathbf{F}_p^0,\end{gathered} \tag{8}$$

where $\mathbf{W}_p^0, \mathbf{W}_p^a, \mathbf{b}_p^0, \mathbf{b}_p^a$ are trainable weight matrix, $\mathbf{F}_p^0$ is the representation of propagation path p and $\mathbf{F}_U^m$ is final user representation. To learn the users' interests, we optimize the model with the loss:

$$\mathcal{L}_{rec} = \sum_{(u,i)\in\mathcal{O}} \|\mathbf{F}_u^m\mathbf{F}_i^{mT} - \mathbf{Y}_{ui}^m\|^2, \tag{9}$$

where $\mathbf{Y}_{ui}^m$ is recent rating matrix from daily user-item behavior network G_o^m.

Besides, due to the information asymmetry issue, we introduce contrastive learning as a regularizer to enrich users' propagated path representations. For this purpose, we maximize the mutual information between one user's path representation $\mathbf{F}_{p,u_i}^0$ and one of his randomly sampled neighbor's path representations $\mathbf{F}_{q,u_j}^0$. In such a way, the path representations in sparse domains can be enriched. In particular, we set as the embedding pair $(\mathbf{F}_{p,u_i}^0, \mathbf{F}_{q,u_j}^0)$ with $(u_i, u_j) \in G_s$ where u_i and u_j are reliable neighbors selected by Gumbel attention in Equation 5 as positive sample. And we generate negative sample $(\mathbf{F}_{p,u_i}^0, \mathbf{F}_{q,u_k}^0)$ by randomly selecting L users $\{u_{k_c}, c = 0, 1, ..., L-1\}$ with $(u_i, u_{k_c}) \notin G_s$. Then we define the mutual information by InfoNCE [5] is as follows:

$$\mathcal{L}_{infoNCE} = \sum_{(u_i,u_j)\in G_s} -\log \frac{\exp\left(\mathbf{F}_{p,u_i}^0 \mathbf{F}_{q,u_j}^{0T}\right)}{\exp\left(\mathbf{F}_{p,u_i}^0 \mathbf{F}_{q,u_j}^{0T}\right) + \sum_{(u_i,u_{k_c})\notin G_s} \exp\left(\mathbf{F}_{p,u_i}^0 \mathbf{F}_{q,u_{k_c}}^{0T}\right)}. \tag{10}$$

Then the total loss of the second stage is the combination of two losses,

$$\mathcal{L}_2 = \mathcal{L}_{rec} + \gamma \mathcal{L}_{infoNCE}, \tag{11}$$

where γ is the coefficient to control the balance of the two losses.

3.6 Complexity Analysis

In this section, we discuss the complexity of our model. In the first stage, the most time-consuming part is graph convolution and Gumbel attention. The time complexity is $\mathcal{O}((|\mathbf{A}_s|+|\mathbf{A}_o|)dK+|\mathbf{Y}_{UI}|d)$. It is easy to see that the first stage is linear to the number of all edges which is the same as traditional GNN models. However, it can be pre-trained only once so the online training process will not have this time cost. So we focus more on the second online daily update stage. There is no graph propagation process which is very time-consuming in the second stage. The training time complexity is $\mathcal{O}(|\mathbf{Y}_{UI}^m|Pd)$ where $|\mathbf{Y}_{UI}^m|$ denotes the number of training user-item pairs in the second stage. And P means the number of paths which is a constant. Therefore, the time complexity of the second online stage is close to a multi-layer MLP model and much lower than that of previous GNN-based social recommendation methods.

4 Experiment

In this section, we conduct experiments to evaluate the performance of SSR. Specifically, we aim to answer the following research questions:

- **RQ1**: How does SSR perform compared to the state-of-the-art baselines?
- **RQ2**: How time efficient is the proposed scalable two-stage framework?
- **RQ3**: Can the proposed SSR be scale to real industry massive datasets?
- **RQ4**: How does each of the key components of SSR affect the model?

Table 1. Dataset Statistics

Dataset	#User	#Item	#Feedback	#Density	#Relation
LastFM	1,892	17,632	92,834	0.28%	25,434
Yelp	17,237	38,342	204,448	0.04%	143,765
Flick	8,358	82,120	314,809	0.05%	187,273
Industry	5,183,534	5,433	22,914,019	0.08%	366,640,997

4.1 Datasets and Baselines

In order to comprehensively evaluate the effectiveness of our proposed method, we evaluate the proposed SSR on three common-used public datasets (i.e., Last.fm[5], Yelp[6], Flickr[7]) and one massive real-world industry recommendation dataset(i.e., Industry). Since there is no time logs in the three public datasets, we use the same user-item interaction in both offline and online stage. In this industry setting, we use user-item interaction of the past month to train the first pre-train stage and user-item interaction of one recent week to train the second online daily-updated stage. The detailed descriptions of the four datasets are summarized in Table 1. Following [15], to perform the evaluation, for each user, we randomly select 1000 unrated items that a user has not interacted with as negative samples, followed by the ranking procedure with the positive samples among 1000 negative samples. Two relevancy-based metrics (i.e., *Precision@15* and *Recall@15*) and one ranking-based metric (i.e., *NDCG@15*) are used to evaluate the performance of all methods. To reduce the uncertainty in this process, we repeat this procedure 5 times and report the average results.

We compare SSR with a set of commonly-used social recommendation baselines, including MF-based and GNN-based models. In summary, **BPR** [12] and **FM** [11] are traditional recommendation methods. **LightGCN** [3] is a fast GNN model. **DiffNet++** [15], **ESRF** [19], **MHCN** [20], **SEPT** [18] are GNN-based social recommendation models. For the general settings of all the methods, we empirically set the dimension of latent embeddings to 64, the balance coefficient γ to 0.01, and the batch size to 2000. We use the Adam optimizer for all these models with an initial learning rate of 0.001. For GNN-based models, the number of graph convolutional layers are set as 2.

4.2 Overall Performance Comparison (RQ1)

Since most of the datasets would take too much time on the largest dataset Industry, we conduct the performance comparison over all baselines on public benchmarks, i.e., Last.fm, Yelp, and Flickr. The results are presented in Table 2. From the results, we have the following observations:

– It is easy to see that our proposed SSR achieves significant improvements over the baselines on all datasets for most cases. Even though our method gets some little negative gains on some datasets with MHCN, the MHCN is much more time-consuming and can not scale to massive industry datasets. The actual training time comparison can be found in the next section.

[5] http://files.grouplens.org/datasets/hetrec2011/
[6] https://www.yelp.com/dataset/challenge
[7] https://www.flickr.com/

Table 2. Recommendation Performance Comparison

Dataset	Metric	BPR	FM	LightGCN	DiffNet++	ESRF	MHCN	SEPT	SSR
Last.fm	Prec@15	0.328	0.312	0.309	0.338	0.339	**0.361**	0.327	0.353
	Recall@15	0.499	0.475	0.469	0.514	0.516	**0.549**	0.497	0.536
	NDCG@15	0.494	0.480	0.476	0.514	0.518	**0.548**	0.479	0.536
Yelp	Prec@15	0.041	0.040	0.038	0.045	0.045	0.045	0.042	**0.047**
	Recall@15	0.372	0.362	0.345	0.412	0.410	0.410	0.377	**0.426**
	NDCG@15	0.208	0.202	0.188	0.226	0.226	0.224	0.206	**0.236**
Flickr	Prec@15	0.052	0.051	0.054	0.054	0.052	0.060	0.055	**0.060**
	Recall@15	0.183	0.174	0.173	0.186	0.186	**0.223**	0.201	0.221
	NDCG@15	0.132	0.128	0.126	0.133	0.133	**0.159**	0.143	0.155

– The performance of our SSR is much better in the more sparse datasets like Yelp and Flickr. It is reasonable because our SSR enriches the sparse domain by self-supervised learning.

4.3 Efficiency Analysis on Benchmarks (RQ2)

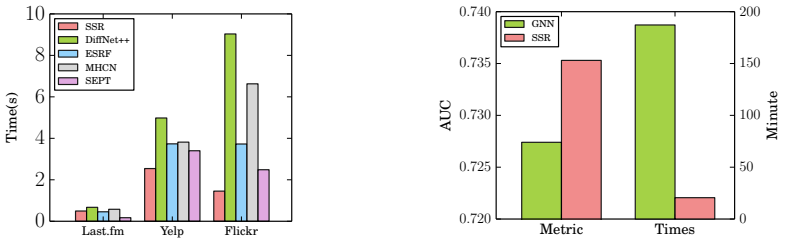

Fig. 2. Left: the training time of social recommendation methods on public benchmarks; Right: the performance and training time on large-scale industry dataset.

We plot the training time per epoch in Figure 2 left, where only the performance of social recommendation models are shown since only these model uses both social network and recommendation network. In the smallest dataset Last.fm, the training time of all methods including our model and baselines are very close. However, our training time is significantly reduced when the datasets become larger (i.e., Yelp and Flickr). And Diffnet++ and MHCN are two of the most time-consuming methods. It demonstrates the scalability of SSR.

4.4 Performance and Efficiency Analysis on Industry Dataset (RQ3)

In this section, we will demonstrate SSR's ability to handle real industry datasets Industry with ten million scale. For the time limitation, we only run a base graph attention-based GNN on a social recommendation as baselines. To make the GNN

runnable on this dataset, we randomly sample ten neighbors for each node like GraphSage [2]. Since the first part of our model is pre-processed offline and only the second part of our model is updated daily, we only count the time of the online part of our model. The results is shown in Figure 2 right. We can see SSR achieves better performance than the baseline GNN in the left bar graph. Besides, the GNN takes much more time (about 9 times) than SSR in the right bar graph.

4.5 Ablation Analysis (RQ4)

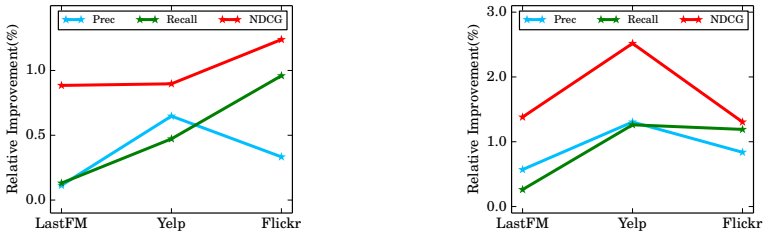

Fig. 3. Left: the relative improvement of SSR over SSR w/o Gumbel attention; Right: the relative improvement of SSR over SSR w/o self-supervised part.

Impact of social selection by Gumbel attention. The effect of social selection by Gumbel attention is shown in Figure 3 left. We change SSR into a variant, **Adj**, which changes the Gumbel attention into a normal adjacency matrix in the first stage. It measures the effect of denoising in social networks. In the figures, we plot the relative improvement over the variants. We can see that SSR performs the best among the variants on all datasets. It demonstrates that the selection of noisy social relations is necessary for social recommendations.

Impact of the self-supervised module. The effect of the self-supervised module is shown in Figure 3 right. We change SSR into a variant, **-InfoNCE** which deletes the infoNCE loss in the second stage. It measure the importance of enriching the sparse domain by self-supervised contrastive learning. The SSR performs consistently better than the variant. It also proves the importance of the self-supervised module.

5 Conclusion

In this paper, we propose a scalable two-stage social recommendation framework named SSR. We decouple our model into two stages, Gumbel-based feature propagation and self-supervised multi-representation, for offline pre-training and online daily updating respectively. In the first stage, we pre-train a model by Gumbel attention to learn the importance of neighbors and propagate the features with multiple propagation layers. In the second stage, we fuse the pre-processed features and use contrastive learning to enrich the representability. Extensive experimental results demonstrate the scalability and effectiveness of our framework.

References

1. Defferrard, M., Bresson, X., Vandergheynst, P.: Convolutional neural networks on graphs with fast localized spectral filtering. NIPS **29** (2016)
2. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. Advances in neural information processing systems **30** (2017)
3. He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., Wang, M.: Lightgcn: Simplifying and powering graph convolution network for recommendation. In: Proceedings of the 43rd International ACM SIGIR conference. pp. 639–648 (2020)
4. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. International Conference on Learning Representations (ICLR) (2017)
5. Liu, X., Zhang, F., Hou, Z., Mian, L., Wang, Z., Zhang, J., Tang, J.: Self-supervised learning: Generative or contrastive. IEEE TKDE (2021)
6. Long, X., Huang, C., Xu, Y., Xu, H., Dai, P., Xia, L., Bo, L.: Social recommendation with self-supervised metagraph informax network. In: Proceedings of the 30th CIKM. pp. 1160–1169 (2021)
7. Lorberbom, G., Johnson, D., Maddison, C.J., Tarlow, D., Hazan, T.: Learning generalized gumbel-max causal mechanisms. NIPS **34** (2021)
8. Ma, H., Yang, H., Lyu, M.R., King, I.: Sorec: social recommendation using probabilistic matrix factorization. In: CIKM. pp. 931–940 (2008)
9. Ma, H., Zhou, D., Liu, C., Lyu, M.R., King, I.: Recommender systems with social regularization. In: WSDM. pp. 287–296 (2011)
10. McPherson, M., Smith-Lovin, L., Cook, J.M.: Birds of a feather: Homophily in social networks. Annual review of sociology **27**(1), 415–444 (2001)
11. Rendle, S.: Factorization machines. In: 2010 IEEE International conference on data mining. pp. 995–1000. IEEE (2010)
12. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: Bpr: Bayesian personalized ranking from implicit feedback. arXiv preprint arXiv:1205.2618 (2012)
13. Rossi, E., Frasca, F., Chamberlain, B., Eynard, D., Bronstein, M., Monti, F.: Sign: Scalable inception graph neural networks. arXiv preprint arXiv:2004.11198 (2020)
14. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph Attention Networks. International Conference on Learning Representations (2018)
15. Wu, L., Li, J., Sun, P., Hong, R., Ge, Y., Wang, M.: Diffnet++: A neural influence and interest diffusion network for social recommendation. TKDE (2020)
16. Xia, X., Yin, H., Yu, J., Shao, Y., Cui, L.: Self-supervised graph co-training for session-based recommendation. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management. pp. 2180–2190 (2021)
17. Yu, J., Gao, M., Yin, H., Li, J., Gao, C., Wang, Q.: Generating reliable friends via adversarial training to improve social recommendation. In: ICDM (2019)
18. Yu, J., Yin, H., Gao, M., Xia, X., Zhang, X., Viet Hung, N.Q.: Socially-aware self-supervised tri-training for recommendation. In: SIGKDD. pp. 2084–2092 (2021)
19. Yu, J., Yin, H., Li, J., Gao, M., Huang, Z., Cui, L.: Enhance social recommendation with adversarial graph convolutional networks. TKDE (2020)
20. Yu, J., Yin, H., Li, J., Wang, Q., Hung, N.Q.V., Zhang, X.: Self-supervised multi-channel hypergraph convolutional network for social recommendation. In: Proceedings of the Web Conference 2021. pp. 413–424 (2021)
21. Yu, L., Shen, J., Li, J., Lerer, A.: Scalable graph neural networks for heterogeneous graphs. arXiv preprint arXiv:2011.09679 (2020)
22. Zhang, Z., Cui, P., Zhu, W.: Deep learning on graphs: A survey. IEEE Transactions on Knowledge and Data Engineering (2020)