Socially-Aware Self-Supervised Tri-Training for Recommendation

Junliang Yu
The University of Queensland
Brisbane, Australia
jl.yu@uq.edu.au

Hongzhi Yin*
The University of Queensland
Brisbane, Australia
h.yin1@uq.edu.au

Min Gao
Chongqing University
Chongqing, China
gaomin@cqu.edu.cn

Xin Xia
The University of Queensland
Brisbane, Australia
x.xia@uq.edu.au

Xiangliang Zhang
KAUST
Thuwal, Saudi Arabia
xiangliang.zhang@kaust.edu.sa

Nguyen Quoc Viet Hung
Griffith University
Gold Coast, Australia
quocviethung1@gmail.com

ABSTRACT

Self-supervised learning (SSL), which can automatically generate ground-truth samples from raw data, holds vast potential to improve recommender systems. Most existing SSL-based methods perturb the raw data graph with uniform node/edge dropout to generate new data views and then conduct the self-discrimination based contrastive learning over different views to learn generalizable representations. Under this scheme, only a bijective mapping is built between nodes in two different views, which means that the self-supervision signals from other nodes are being neglected. Due to the widely observed homophily in recommender systems, we argue that the supervisory signals from other nodes are also highly likely to benefit the representation learning for recommendation. To capture these signals, a general socially-aware SSL framework that integrates tri-training is proposed in this paper. Technically, our framework first augments the user data views with the user social information. And then under the regime of tri-training for multi-view encoding, the framework builds three graph encoders (one for recommendation) upon the augmented views and iteratively improves each encoder with self-supervision signals from other users, generated by the other two encoders. Since the tri-training operates on the augmented views of the same data sources for self-supervision signals, we name it self-supervised tri-training. Extensive experiments on multiple real-world datasets consistently validate the effectiveness of the self-supervised tri-training framework for improving recommendation. The code is released at https://github.com/Coder-Yu/QRec.

KEYWORDS

Self-Supervised Learning, Tri-Training, Recommender Systems, Contrastive Learning

1 INTRODUCTION

Self-supervised learning (SSL) [17], emerging as a novel learning paradigm that does not require human-annotated labels, recently has received considerable attention in a wide range of fields [5, 8, 16, 21, 23, 27, 45]. As the basic idea of SSL is to learn with the automatically generated supervisory signals from the raw data, which is an antidote to the problem of data sparsity in recommender systems, SSL holds vast potential to improve recommendation quality. The recent progress in self-supervised graph representation learning [14, 27, 40] has identified an effective training scheme for graph-based tasks. That is, performing stochastic augmentation by perturbing the raw graph with uniform node/edge dropout or random feature shuffling/masking to create supplementary views and then maximizing the agreement between the representations of the same node but learned from different views, which is known as *graph contrastive learning* [40]. Inspired by its effectiveness, a few studies [19, 29, 37, 46] then follow this training scheme and are devoted to transplanting it to recommendation.

With these research effort, the field of self-supervised recommendation recently has demonstrated some promising results showing that mining supervisory signals from stochastic augmentations is desirable [29, 46]. However, in contrast to other graph-based tasks, recommendation is distinct because there is widely observed *homophily* across users and items [20]. Most existing SSL-based methods conduct the self-discrimination based contrastive learning over the augmented views to learn generalizable representations against the variance in the raw data. Under this scheme, a bijective mapping is built between nodes in two different views, and a given node can just exploit information from itself in another view. Meanwhile, the other nodes are regarded as the negatives that are pushed apart from the given node in the latent space. Obviously, a number of nodes are false negatives which are similar to the given node due to the homophily, and can actually benefit representation learning in the scenario of recommendation if they are recognized as the positives. Conversely, roughly classifying them into the negatives could lead to a performance drop.

To tackle this issue, a socially-aware SSL framework which combines the tri-training [47] (multi-view co-training) with SSL is proposed in this paper. For supplementary views that can capture the homophily among users, we resort to social relations which can be another data source that implicitly reflects users' preferences [4, 38, 41–43]. Owing to the prevalence of social platforms in the past decade, social relations are now readily accessible in many recommender systems. We exploit the triadic structures in the user-user and user-item interactions to augment two supplementary data views, and socially explain them as profiling users' interests in expanding social circles and sharing desired items to friends, respectively. Given the use-item view which contains users' historical purchases, we have three views that characterize users' preferences

*Corresponding author and having equal contribution with the first author.

from different perspectives and also provide us with a scenario to fuse tri-training and SSL.

Tri-training [47] is a popular semi-supervised learning algorithm which exploits unlabeled data using three classifiers. In this work, we employ it to mine self-supervision signals from other users in recommender systems with the multi-view encoding. Technically, we first build three asymmetric graph encoders over the three views, of which two are only for learning user representations and giving pseudo-labels, and another one working on the user-item view also undertakes the task of generating recommendations. Then we dynamically perturb the social network and user-item interaction graph to create an unlabeled example set. Following the regime of tri-training, during each epoch, the encoders over the other two views predict the most probable semantically positive examples in the unlabeled example set for each user in the current view. Then the framework refines the user representations by maximizing the agreement between representations of labeled users in the current view and the example set through the proposed *neighbor-discrimination* based contrastive learning. As all the encoders iteratively improve in this process, the generated pseudo-labels also become more informative, which in turn recursively benefit the encoders again. The recommendation encoder over the user-item view thus becomes stronger in contrast to those only enhanced by the self-discrimination SSL scheme. So far, there is no work that integrates both tri-training and SSL for graph-based recommendation. Since the tri-training operates on the complementary views of the same data sources to learn self-supervision signals, we name it self-supervised tri-training.

The major contributions of this paper are summarized as follows:

- We propose a general socially-aware self-supervised tri-training framework for recommendation. By unifying the recommendation task and the SSL task under this framework, the recommendation performance can achieve significant gains.
- We propose to exploit positive self-supervision signals from other users and develop a neighbor-discrimination based contrastive learning method.
- We conduct extensive experiments on multiple real-world datasets to demonstrate the advantages of the proposed SSL framework and investigate the effectiveness of every module.

The rest of this paper is structured as follows. Section 2 summarizes the related work of recommendation and SSL. Section 3 introduces the proposed framework. The experimental results are reported in Section 4. Finally, Section 5 concludes this paper.

2 RELATED WORK

2.1 Graph Neural Recommendation Models

Recently, *graph neural networks* (GNNs) [7, 34] have gained considerable attention in the field of recommender systems for their effectiveness in solving graph-related recommendation tasks. Particularly, GCN [15], as the prevalent formulation of GNNs which is a first-order approximation of spectral graph convolutions, has driven a multitude of graph neural recommendation models like GCMC [2], NGCF [28], and LightGCN [11]. The basic idea of these GCN-based models is to exploit the high-order neighbors in the user-item graph by aggregating the embeddings of neighbors to refine the target node's embeddings [33]. In addition to these general models, GNNs also empower other recommendation methods working on specific graphs such as SR-GNN [32] and DHCN [35] over the session-based graph, and DiffNet [31] and MHCN [44] over the social network. It is worth mentioning that GNNs are often used for social computing as the information spreading in social networks can be well captured by the message passing in GNNs [31]. That is the reason why we resort to social networks for self-supervisory signals generated by graph neural encoders.

2.2 Self-Supervised Learning in RS

Self-supervised learning [17] (SSL) is an emerging paradigm to learn with the automatically generated ground-truth samples from the raw data. It was firstly used in visual representation learning and language modeling [1, 5, 10, 12, 45] for model pretraining. The recent progress in SSL seeks to harness this flexible learning paradigm for graph representation learning [22, 23, 26, 27]. SSL models over graphs mainly mine self-supervision signals by exploiting the graph structure. The dominant regime of this line of research is graph contrastive learning which contrasts multiple views of the same graph where the incongruent views are built by conducting stochastic augmentations on the raw graph [9, 23, 27, 40]. The common types of stochastic augmentations include but are not limited to uniform node/edge dropout, random feature/attribute shuffling, and subgraph sampling using random walk.

Inspired by the success of graph contrastive learning, there have been some recent works [19, 29, 37, 46] which transplant the same idea to the scenario of recommendation. Zhou *et al.* [46] devise auxiliary self-supervised objectives by randomly masking attributes of items and skipping items and subsequences of a given sequence for pretraining sequential recommendation model. Yao *et al.* [37] propose a two-tower DNN architecture with uniform feature masking and dropout for self-supervised item recommendation. Ma *et al.* [19] mine extra signals for supervision by looking at the longer-term future and reconstruct the future sequence for self-supervision, which adopts feature masking in essence. Wu *et al.* [29] summarize all the stochastic augmentations on graphs and unify them into a general self-supervised graph learning framework for recommendation. Besides, there are also some studies [25, 36, 44] refining user representations with mutual information maximization among a set of certain members (e.g. ad hoc groups) for self-supervised recommendation. However, these methods are used for specific situations and cannot be easily generalized to other scenarios.

3 PROPOSED FRAMEWORK

In this section, we present our **SE**lf-su**P**ervised **T**ri-training framework, called **SEPT**, with the goal of mining self-supervision signals from other users by the multi-view encoding. The overview of SEPT is illustrated in Fig. 1.

3.1 Preliminaries

3.1.1 Notations. In this paper, we use two graphs as the data sources including the user-item interaction graph $\mathcal{G}_r$ and the user social network $\mathcal{G}_s$. $\mathcal{U} = \{u_1, u_2, ..., u_m\}$ ($|\mathcal{U}| = m$) denotes the user nodes across both $\mathcal{G}_r$ and $\mathcal{G}_s$, and $\mathcal{I} = \{i_1, i_2, ..., i_n\}$ ($|\mathcal{I}| = n$) denotes the item nodes in $\mathcal{G}_r$. As we focus on item recommendation,

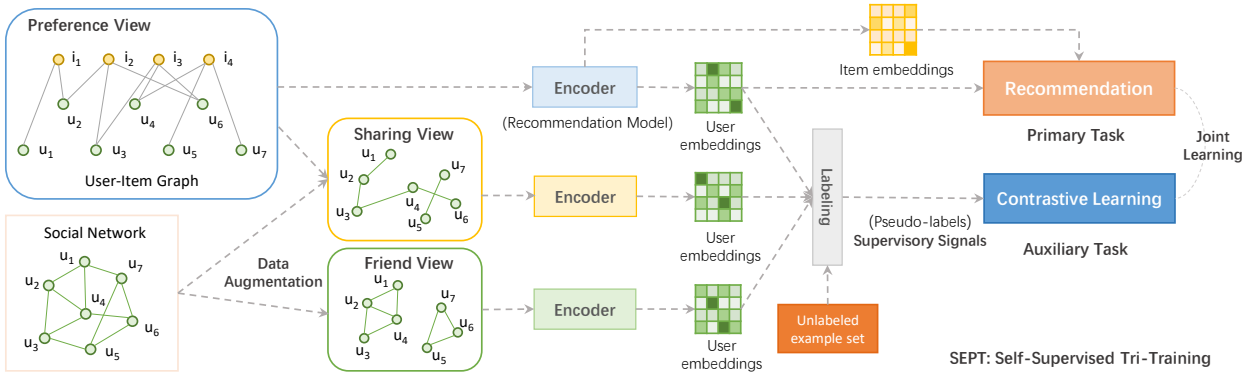

Figure 1: Overview of the proposed self-supervised tri-training framework.

$\boldsymbol{R} \in \mathbb{R}^{m \times n}$ is the binary matrix with entries only 0 and 1 that represent user-item interactions in $\mathcal{G}_r$. For each entry (u, i) in $\boldsymbol{R}$, if user u has consumed/clicked item i, $r_{ui} = 1$, otherwise $r_{ui} = 0$. As for the social relations, we use $\boldsymbol{S} \in \mathbb{R}^{m \times m}$ to denote the social adjacency matrix which is binary and symmetric because we work on undirected social networks with bidirectional relations. We use $\boldsymbol{P} \in \mathbb{R}^{m \times d}$ and $\boldsymbol{Q} \in \mathbb{R}^{n \times d}$ to denote the learned final user and item embeddings for recommendation, respectively. To facilitate the reading, in this paper, matrices appear in bold capital letters and vectors appear in bold lower letters.

3.1.2 Tri-Training. Tri-training [47] is a popular semi-supervised learning algorithm which develops from the *co-training* paradigm [3] and tackles the problem of determining how to label the unlabeled examples to improve the classifiers. In contrast to the standard co-training algorithm which ideally requires two sufficient, redundant and conditionally independent views of the data samples to build two different classifiers, tri-training is easily applied by lifting the restrictions on training sets. It does not assume sufficient redundancy among the data attributes, and initializes three diverse classifiers upon three different data views generated via bootstrap sampling [6]. Then, in the labeling process of tri-training, for any classifier, an unlabeled example can be labeled for it as long as the other two classifiers agree on the labeling of this example. The generated pseudo-label is then used as the ground-truth to train the corresponding classifier in the next round of labeling.

3.2 Data Augmentation

3.2.1 View Augmentation. As has been discussed, there is widely observed homophily in recommender systems. Namely, users and items have many similar counterparts. To capture the homophily for self-supervision, we exploit the user social relations for data augmentation as the social network is often known as a reflection of homophily [20, 39] (i.e., users who have similar preferences are more likely to become connected in the social network and vice versa). Since many service providers such as Yelp[1] encourage users to interact with others on their platforms, it provides their recommender systems with opportunities to leverage abundant social relations. However, as social relations are inherently noisy [41, 43], for accurate supplementary supervisory information, SEPT only utilizes the reliable social relations by exploiting the ubiquitous triadic closure [13] among users. In a socially-ware recommender system, by aligning the user-item interaction graph $\mathcal{G}_r$ and the social network $\mathcal{G}_s$, we can readily get two types of triangles: three users socially connected with each other (e.g. u_1, u_2 and u_4 in Fig. 1) and two socially connected users with the same purchased item (e.g. u_1, u_2 and i_1 in Fig. 1). The former is socially explained as profiling users' interests in expanding social circles, and the latter is characterizing users' interests in sharing desired items with their friends. It is straightforward to regard the triangles as strengthened ties because if two persons in real life have mutual friends or common interests, they are more likely to have a close relationship.

Following our previous work [44], the mentioned two types of triangles can be efficiently extracted in the form of matrix multiplication. Let $\boldsymbol{A}_f \in \mathbb{R}^{m \times m}$ and $\boldsymbol{A}_s \in \mathbb{R}^{m \times m}$ denote the adjacency matrices of the users involved in these two types of triangular relations. They can be calculated by:

$$\boldsymbol{A}_f = (\boldsymbol{SS}) \odot \boldsymbol{S}, \quad \boldsymbol{A}_s = (\boldsymbol{RR}^\top) \odot \boldsymbol{S}. \tag{1}$$

The multiplication $\boldsymbol{SS}$ ($\boldsymbol{RR}^\top$) accumulates the paths connecting two user via shared friends (items), and the Hadamard product $\odot \boldsymbol{S}$ makes these paths into triangles. Since both $\boldsymbol{S}$ and $\boldsymbol{R}$ are sparse matrices, the calculation is not time-consuming. The operation $\odot \boldsymbol{S}$ ensures that the relations in $\boldsymbol{A}_f$ and $\boldsymbol{A}_s$ are subsets of the relations in $\boldsymbol{S}$. As $\boldsymbol{A}_f$ and $\boldsymbol{A}_s$ are not binary matrices, Eq. (1) can be seen a special case of bootstrap sampling on $\boldsymbol{S}$ with the complementary information from $\boldsymbol{R}$. Given $\boldsymbol{A}_f$ and $\boldsymbol{A}_s$ as the augmentation of $\boldsymbol{S}$ and $\boldsymbol{R}$, we have three views that characterize users' preferences from different perspectives and also provide us with a scenario to fuse tri-training and SSL. To facilitate the understanding, we name the view over the user-item interaction graph *preference view*, the view over the triangular social relations *friend view*, and another one *sharing view*, which are represented by $\boldsymbol{R}$, $\boldsymbol{A}_f$, and $\boldsymbol{A}_s$, respectively.

3.2.2 Unlabeled Example Set. To conduct tri-training, an unlabeled example set is required. We follow existing works [29, 40] to perturb the raw graph with edge dropout at a certain probability ρ to create a corrupted graph from where the learned user presentations are

[1] http://www.yelp.com

used as the unlabeled examples. This process can be formulated as:

$$\tilde{\mathcal{G}} = (\mathcal{N}_r \cup \mathcal{N}_s, \boldsymbol{m} \odot (\mathcal{E}_r \cup \mathcal{E}_s)), \tag{2}$$

where $\mathcal{N}_r$ and $\mathcal{N}_s$ are nodes, $\mathcal{E}_r$ and $\mathcal{E}_s$ are edges in $\mathcal{G}_r$ and $\mathcal{G}_s$, and $\boldsymbol{m} \in \{0,1\}^{|\mathcal{E}_r \cup \mathcal{E}_s|}$ is the masking vector to drop edges. Herein we perturb both $\mathcal{G}_r$ and $\mathcal{G}_s$ instead of $\mathcal{G}_r$ only, because the social information is included in the aforementioned two augmented views. For integrated self-supervision signals, perturbing the joint graph is necessary.

3.3 SEPT: Self-Supervised Tri-Training

3.3.1 Architecture. With the augmented views and the unlabeled example set, we follow the setting of tri-training to build three encoders. Architecturally, the proposed self-supervised training framework can be model-agnostic so as to boost a multitude of graph neural recommendation models. But for a concrete framework which can be easily followed, we adopt LightGCN [11] as the basic structure of the encoders due to its simplicity. The general form of encoders is defined as follows:

$$\boldsymbol{Z} = H(\boldsymbol{E}, \mathcal{V}), \tag{3}$$

where H is the encoder, $\boldsymbol{Z} \in \mathbb{R}^{m \times d}$ or $\mathbb{R}^{(m+n) \times d}$ denotes the final representation of nodes, $\boldsymbol{E}$ of the same size denotes the initial node embeddings which are the bottom shared by the three encoders, and $\mathcal{V} \in \{\boldsymbol{R}, \boldsymbol{A}_s, \boldsymbol{A}_f\}$ is any of the three views. It should be noted that, unlike the vanilla tri-training, SEPT is asymmetric. The two encoders H_f and H_s that work on the *friend view* and *sharing view* are only in charge of learning user representations through graph convolution and giving pseudo-labels, while the encoder H_r working on the *preference view* also undertakes the task of generating recommendations and thus learns both user and item representations (shown in Fig. 1). Let H_r be the dominant encoder (recommendation model), and H_f and H_s be the auxiliary encoders. Theoretically, given a concrete H_r like LightGCN [11], there should be the optimal structures of H_f and H_s. However, exploring the optimal structures of the auxiliary encoders is out of the scope of this paper. For simplicity, we assign the same structure to H_f and H_s. Besides, to learn representations of the unlabeled examples from the perturbed graph $\tilde{\mathcal{G}}$, another encoder is required, but it is only for graph convolution. All the encoders share the bottom embeddings $\boldsymbol{E}$ and are built over different views with the LightGCN structure.

3.3.2 Constructing Self-Supervision Signals. By performing graph convolution over the three views, the encoders learn three groups of user representations. As each view reflects a different aspect of the user preference, it is natural to seek supervisory information from the other two views to improve the encoder of the current view. Given a user, we predict its semantically positive examples in the unlabeled example set using the user representations from the other two views. Taking user u in the preference view as an instance, the labeling is formulated as:

$$\begin{aligned} &\boldsymbol{x}_u^s = \tilde{\boldsymbol{Z}} \boldsymbol{z}_u^s, \quad \boldsymbol{x}_u^f = \tilde{\boldsymbol{Z}} \boldsymbol{z}_u^f, \\ &\boldsymbol{y}_{u+}^s = Softmax(\phi(\boldsymbol{x}_u^s)), \quad \boldsymbol{y}_{u+}^f = Softmax(\phi(\boldsymbol{x}_u^f)), \end{aligned} \tag{4}$$

where ϕ is the *cosine* operation, $\boldsymbol{z}_u^s$ and $\boldsymbol{z}_u^f$ are the representations of user u learned by H_s and H_f, respectively, $\tilde{\boldsymbol{Z}}$ is the representations of users in the unlabeled example set obtained through graph convolution, and $\boldsymbol{y}_{u+}^s$ and $\boldsymbol{y}_{u+}^f$ denote the predicted probability of each user being the semantically positive example of user u in the corresponding views.

Under the scheme of tri-training, to avoid noisy examples, only if both H_s and H_f agree on the labeling of a user being the positive sample, and then the user can be labeled for H_r. We obey this rule and add up the predicted probabilities from the two views and obtain:

$$\boldsymbol{y}_{u+}^r = \frac{1}{2}(\boldsymbol{y}_{u+}^s + \boldsymbol{y}_{u+}^f). \tag{5}$$

With the probabilities, we can select K positive samples with the highest confidence. This process can be formulated as:

$$\mathcal{P}_{u+}^r = \{\tilde{\boldsymbol{Z}}_k \mid k \in \text{Top-}K(\boldsymbol{y}_{u+}^r),\ \tilde{\boldsymbol{Z}} \sim \tilde{\mathcal{G}}\}. \tag{6}$$

In each iteration, $\tilde{\mathcal{G}}$ is reconstructed with the random edge dropout for varying user representations. SEPT dynamically generates positive pseudo-labels over this data augmentation for each user in every view. Then these labels are used as the supervisory signals to refine the shared bottom representations.

3.3.3 Contrastive Learning. Having the generated pseudo-labels, we develop the *neighbor-discrimination* contrastive learning method to fulfill self-supervision in SEPT.

Given a certain user, we encourage the consistency between his node representation and the labeled user representations from $\mathcal{P}_{u+}$, and minimizing the agreement between his representation and the unlabeled user representations. The idea of the neighbor-discrimination is that, given a certain user in the current view, the positive pseudo-labels semantically represent his neighbors or potential neighbors in the other two views, then we should also bring these positive pairs together in the current view due to the homophily across different views. And this can be achieved through the *neighbor-discrimination* contrastive learning. Formally, we follow the previous studies [5, 29] to adopt InfoNCE [12], which is effective in mutual information estimation, as our learning objective to maximize the agreement between positive pairs and minimize that of negative pairs:

$$\mathcal{L}_{ssl} = -\mathbb{E} \sum_{v \in \{r,s,f\}} \left[\log \frac{\sum_{p \in \mathcal{P}_{u+}^v} \psi(z_u^v, \tilde{z}_p)}{\sum_{p \in \mathcal{P}_{u+}^v} \psi(z_u^v, \tilde{z}_p) + \sum_{j \in U / \mathcal{P}_{u+}^v} \psi(z_u^v, \tilde{z}_j)} \right] \tag{7}$$

where $\psi(z_u^v, \tilde{z}_p) = \exp\left(\phi(z_u^v \cdot \tilde{z}_p)/\tau\right)$, $\phi(\cdot) : \mathbb{R}^d \times \mathbb{R}^d \longmapsto \mathbb{R}$ is the discriminator function that takes two vectors as the input and then scores the agreement between them, and τ is the temperature to amplify the effect of discrimination ($\tau = 0.1$ is the best in our implementation). We simply implement the discriminator by applying the cosine operation. Compared with the self-discrimination, the neighbor-discrimination leverages the supervisory signals from the other users. When only one positive example is used and if the user itself in $\tilde{Z}$ has the highest confidence in $\boldsymbol{y}_{u+}$, the neighbor-discrimination degenerates to the self-discrimination. So, the self-discrimination can be seen as a special case of the neighbor-discrimination. But when a sufficient number of positive examples are used, these two methods could also be simultaneously

adopted because the user itself in $\tilde{Z}$ is often highly likely to be in the Top-K similar examples $\mathcal{P}_{u+}$. With the training proceeding, the encoders iteratively improve to generate evolving pseudo-labels, which in turn recursively benefit the encoders again.

Compared with the vanilla tri-training, it is worth noting that in SEPT, we do not add the pseudo-labels into the adjacency matrices for subsequent graph convolution during training. Instead, we adopt a soft and flexible way to guide the user representations via mutual information maximization, which is distinct from the vanilla tri-training that adds the pseudo-labels to the training set for next-round training. The benefits of this modeling are two-fold. Firstly, adding pseudo-labels leads to reconstruction of the adjacency matrices after each iteration, which is time-consuming; secondly, the pseudo-labels generated at the early stage might not be informative; repeatedly using them would mislead the framework.

3.3.4 Optimization. The learning of SEPT consists of two tasks: recommendation and the neighbor-discrimination based contrastive learning. Let $\mathcal{L}_r$ be the BPR pairwise loss function [24] which is defined as:

$$\mathcal{L}_r = \sum_{i \in \mathcal{I}(u), j \notin \mathcal{I}(u)} -\log \sigma(\hat{r}_{ui} - \hat{r}_{uj}) + \lambda \|\boldsymbol{E}\|_2^2, \tag{8}$$

where $\mathcal{I}(u)$ is the item set that user u has interacted with, $\hat{r}_{ui} = \boldsymbol{P}_u^\top \boldsymbol{Q}_i$, $\boldsymbol{P}$ and $\boldsymbol{Q}$ are obtained by splitting $\boldsymbol{Z}^r$, and λ is the coefficient controlling the L_2 regularization. The training of SEPT proceeds in two stages: *initialization* and *joint learning*. To start with, we warm up the framework with the recommendation task by optimizing $\mathcal{L}_r$. Once trained with $\mathcal{L}_r$, the shared bottom $\boldsymbol{E}$ has gained far stronger representations than randomly initialized embeddings. The self-supervised tri-training then proceeds as described in Eq. (4) - (7), acting as an auxiliary task which is unified into a joint learning objective to enhance the performance of the recommendation task. The overall objective of the joint learning is defined as:

$$\mathcal{L} = \mathcal{L}_r + \beta \mathcal{L}_{ssl}, \tag{9}$$

where β is a hyper-parameter used to control the magnitude of the self-supervised tri-training. The overall process of SEPT is presented in Algorithm 1.

Algorithm 1: The running process of SEPT

Input: Bidirectional social relations $\mathbf{S}$, User feedback $\mathbf{R}$, and randomly initialized node embeddings $\boldsymbol{E}$;
Output: Recommendation lists
Pretraining with $\mathcal{L}_r$ in Eq. (8);
View augmentation with Eq. (1);
for *each iteraction* **do**
  Construct $\tilde{\mathcal{G}}$ and obtain the unlabeled example set through graph convolution;
  for *each batch* **do**
    Randomly select c users from $\tilde{Z}$ to be labeled;
    for *each user* u **do**
      Predict the probabilities of the c users being the semantically positive examples in different views with Eq. (4) - (5);
      Obtain Top-K positive examples with Eq. (6);
    end
    Jointly optimize the overall objective in Eq. (9);
  end
end

3.4 Discussions

3.4.1 Connection with Social Regularization. Social recommendation [38, 43, 44] integrates social relations into recommender systems to address the data sparsity issue. A common idea of social recommendation is to regularize user representations by minimizing the euclidean distance between socially connected users, which is termed *social regularization* [18]. Although the proposed SEPT also leverages socially-aware supervisory signals to refine user representations, it is distinct from the social regularization. The differences are also two-fold. Firstly, the social regularization is a static process which is always performed on the socially connected users, whereas the neighbor-discrimination is dynamic and iteratively improves the supervisory signals imposed on uncertain users; secondly, negative social relations (dislike) cannot be readily retrieved in social recommendation, and hence the social regularization can only keep socially connected users close. But SEPT can also pushes users who are not semantically positive in the three views apart.

3.4.2 Complexity. Architecturally, SEPT can be model-agnostic, and its complexity mainly depends on the structure of the used encoders. In this paper, we present a LightGCN-based architecture. Given $O(|\boldsymbol{R}|d)$ as the time complexity of the recommendation encoder for graph convolution, the total complexity for the graph convolution is less than $3O(|\boldsymbol{R}|d)$ because $\boldsymbol{A}_f$ and $\boldsymbol{A}_s$ are usually sparser than $\boldsymbol{R}$. The prime cost of the labeling process comes from the Top-K operation in Eq. (6), which usually requires $O(m \log(K))$ by using the max heap. To reduce the cost and speed up training, in each batch for training, only c ($c \ll m$, e.g. 1000) users in a batch are randomly selected and being the unlabeled example set of the pseudo-labels, and this sampling method can also prevent overfitting. The complexity of the neighbor-discrimination based contrastive learning is $O(cd)$.

Table 1: Dataset Statistics

Dataset	#User	#Item	#Feedback	#Relation	Density
Last.fm	1,892	17,632	92,834	25,434	0.28%
Douban-Book	13,024	22,347	792,062	169,150	0.27%
Yelp	19,539	21,266	450,884	363,672	0.11%

4 EXPERIMENTAL RESULTS

4.1 Experimental Settings

Datasets. Three real-world datasets: Last.fm[2], Douban-Book[3], and Yelp [4] are used in our experiments to evaluate SEPT. As SEPT aims to improve Top-N recommendation, we follow the convention in

[2] http://files.grouplens.org/datasets/hetrec2011/
[3] https://github.com/librahu/HIN-Datasets-for-Recommendation-and-Network-Embedding
[4] https://github.com/Coder-Yu/QRec

Table 2: Performance improvements brought by SEPT on the three datasets.

Method		Last.fm			Douban-Book			Yelp		
		Prec@10	Rec@10	NDCG@10	Prec@10	Rec@10	NDCG@10	Prec@10	Rec@10	NDCG@10
1-Layer	LightGCN	17.517	17.463	21.226	6.712	8.248	9.584	2.433	6.083	4.688
	SEPT	18.266	18.503	21.985	6.931	8.984	10.116	2.620	6.738	5.233
		↑ 4.275%	↑ 5.955%	↑ 3.575%	↑ 3.262%	↑ 8.923%	↑ 5.551%	↑ 7.685%	↑ 10.767%	↑ 11.625%
2-Layer	LightGCN	19.205	19.480	23.392	7.650	10.024	11.348	2.641	6.791	5.235
	SEPT	20.191	20.488	24.507	8.383	10.810	12.635	2.901	7.460	5.776
		↑ 5.134%	↑ 5.174%	↑ 4.766%	↑ 9.582%	↑ 7.841%	↑ 11.341%	↑ 9.844%	↑ 9.851%	↑ 10.334%
3-Layer	LightGCN	18.827	19.234	22.904	7.275	9.599	10.836	2.426	6.130	4.703
	SEPT	19.082	19.363	23.175	7.868	10.458	11.928	2.635	6.807	5.250
		↑ 1.354%	↑ 0.671%	↑ 1.183%	↑ 8.151%	↑ 8.948%	↑ 10.077%	↑ 8.615%	↑ 11.044%	↑ 11.631%

previous research [43, 44] to leave out ratings less than 4 in the dataset of Douban-Book which consists of explicit ratings with a 1-5 rating scale, and assign 1 to the rest. The statistics of the datasets is shown in Table 1. For precise assessment, 5-fold cross-validation is conducted in all the experiments and the average results are presented.

Baselines. Three recent graph neural recommendation models are compared with SEPT to test the effectiveness of the self-supervised tri-training for recommendation:

- **LightGCN** [11] is a GCN-based general recommendation model that leverages the user-item proximity to learn node representations and generate recommendations, which is reported as the state-of-the-art.
- **DiffNet++** [30] is a recent GCN-based social recommendation method that models the recursive dynamic social diffusion in both the user and item spaces.
- **MHCN** [44] is a latest hypergraph convolutional network-based social recommendation method that models the complex correlations among users with hyperedges to improve recommendation performance.

LightGCN [11] is the basic encoder in SEPT. Investigating the performance of LightGCN and SEPT is essential. Since LightGCN is a widely acknowledged SOTA baseline reported in many recent papers [29, 44], we do not compare SEPT with other weak baselines such as NGCF [28], GCMC [2], and BPR [24]. Two strong social recommendation models are also compared to SEPT to verify that the self-supervised tri-training, rather than the use of social relations, is the main driving force of the performance improvements.

Metrics. To evaluate all the methods, we first perform item ranking on all the candidate items. Then two relevancy-based metrics *Precision@10* and *Recall@10* and one ranking-based metric *NDCG@10* are calculated on the truncated recommendation lists, and the values are presented in percentage.

Settings. For a fair comparison, we refer to the best parameter settings reported in the original papers of the baselines and then fine tune all the hyperparameters of the baselines to ensure the best performance of them. As for the general settings of all the methods, we empirically set the dimension of latent factors (embeddings) to 50, the regularization parameter λ to 0.001, and the batch size to 2000. In section 4.4, we investigate the parameter sensitivity of SEPT, and the best parameters are used in section 4.2 and 4.3. We use Adam to optimize all these models with an initial learning rate 0.001.

4.2 Overall Performance Comparison

In this part, we validate if SEPT can improve recommendation. The performance comparisons are shown in Table 2 and 3. We conduct experiments with different layer numbers in Table 2. In Table 3, a two-layer setting is adopted for all the methods because they all reach their best performance on the used datasets under this setting. The performance improvement (drop) marked by ↑ (↓) is calculated by using the performance difference to divide the subtrahend. According to the results, we can draw the following observations and conclusions:

- Under all the different layer settings, SEPT can significantly boost LightGCN. Particularly, on the sparser datasets: Douban-Book and Yelp, the improvements get higher. The maximum improvement can even reach 11%. This can be an evidence that demonstrates the effectiveness of self-supervised learning. Besides, although both LightGCN and SEPT suffer the over-smoothed problem when the layer number is 3, SEPT can still outperform LightGCN. We think the possible reason is that contrastive learning can, to some degree, alleviate the over-smooth problem because the dynamically generated unlabeled examples provide sufficient data variance.

In addition to the comparison with LightGCN, we also compare SEPT with social recommendation models to validate if the self-supervised tri-training rather than social relations primarily promote the recommendation performance. Since MHCN is also built upon LightGCN, comparing these two models can be more informative. Besides, S^2-MHCN, which is the self-supervised variant of MHCN is also compared. The improvements (drops) are calculated by comparing the results of SEPT and S^2-MHCN. According to the results in Table 3, we make the following observations and conclusions:

- Although integrating social relations into graph neural models are helpful (comparing MHCN with LightGCN), learning under the scheme of SEPT can achieve more performance gains (comparing SEPT with MHCN). DiffNet++ is uncompetitive compared with the other three methods. Its failure can be attributed to its

Table 3: Performance comparison with social recommendation models on three datasets.

Method	Last.fm			Douban-Book			Yelp		
	Prec@10	Rec@10	NDCG@10	Prec@10	Rec@10	NDCG@10	Prec@10	Rec@10	NDCG@10
DiffNet++	18.485	18.737	22.310	7.250	9.511	9.591	2.480	6.354	4.833
MHCN	19.625	19.945	23.834	7.718	10.113	11.540	2.751	6.862	5.356
S^2-MHCN	20.052	20.375	24.395	8.083	10.402	12.136	3.003	7.885	6.061
SEPT	20.191	20.488	24.507	8.383	10.810	12.635	2.901	7.460	5.776
	↑ 0.693%	↑ 0.554%	↑ 0.459%	↑ 3.711%	↑ 3.922%	↑ 4.111%	↓ 3.396%	↓ 5.389%	↓ 4.702%

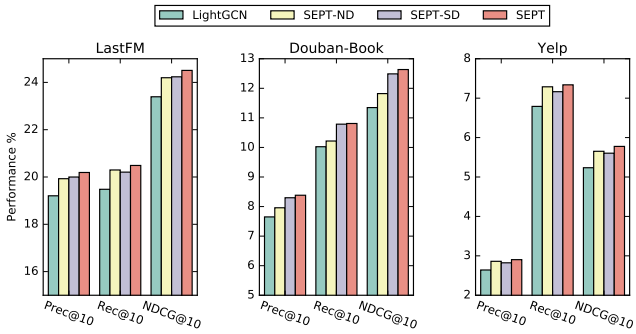

Figure 2: Comparisons between self-discrimination and neighbor-discrimination.

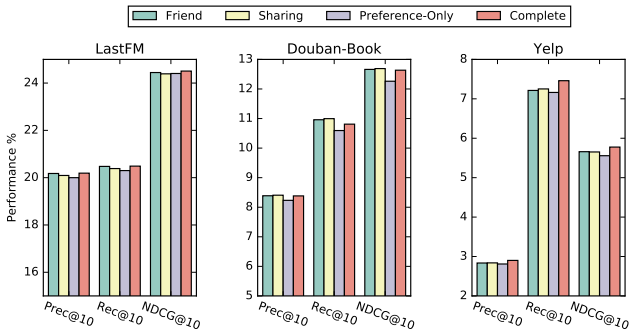

Figure 3: Contributions of each component in SEPT.

redundant and useless parameters and operations [11]. On both LastFM and Douban-Book, SEPT outperforms S^2-MHCN. On Yelp, S^2-MHCN exhibits better performance than SEPT does. The superiority of SEPT and S^2-MHCN demonstrates that self-supervised learning holds vast capability for improving recommendation. In addition, SEPT does not need to learn other parameters except the bottom embeddings, whereas there are a number of other parameters that S^2-MHCN needs to learn. Meanwhile, SEPT runs much faster than S^2-MHCN does in our experiments, which makes it more competitive even that it is beaten by S^2-MHCN on Yelp by a small margin.

4.3 Self-Discrimination *v.s.* Neighbor-Discrimination

In SEPT, the generated positive examples can include both the user itself and other users in the unlabeled example set. It is not clear which part contributes more to the recommendation performance. In this part, we investigate the self-discrimination and the neighbor-discrimination without the user itself being the positive example. For convenience, we use SEPT-SD to denote the self-discrimination, and SEPT-ND to denote the latter. It also should be mentioned that, for SEPT-ND only, a small $\beta = 0.001$ can lead to the best performance on all the datasets. A two-layer setting is used in this case.

According to Fig. 2, we can observe that both SEPT-SD and SEPT-ND exhibit better performances than LightGCN does, which proves that both the supervisory signals from the user itself and other users can benefit a self-supervised recommendation model. Our claim about the self-supervision signals from other users is validated. Besides, the importance of the self-discrimination and the neighbor-discrimination varies from dataset to dataset. On LastFM, they almost contribute equally. On Douban-Book, self-discrimination shows much more importance. On Yelp, neighbor-discrimination is more effective. This phenomenon can be explained by Fig. 5. With the increase of the used positive examples, we see that the performance of SEPT almost remains stable on LastFM and Yelp but gradually declines on Douban-Book. We guess that there is widely observed homophily in LastFM and Yelp, so a large number of users share similar preferences, which can be the high-quality positive examples in these two datasets. However, users in Douban-Book may have more diverse interests, which results in the quality drop when the number of used positive examples increases.

4.4 View Study

In SEPT, we build two augmented views to conduct tri-training for mining supervisory signals. In this part, we ablate the framework to investigate the contribution of each view. A two-layer setting is used in this case. In Fig. 3, 'Friend' or 'Sharing' means that the corresponding view is detached. When only two views are used, SEPT degenerates to the self-supervised co-training. 'Preference-Only' means that only the preference view is used. In this case, SEPT further degenerates to the self-training.

From Fig. 3, we can observe that on both LastFM and Yelp, all the views contribute, whereas on Douban-Book, the self-supervised co-training setting achieves the best performance. Moreover, when only the preference view is used, SEPT shows lower performance but it is still better than that of LightGCN. With the decrease of used number of views, the performance of SEPT slightly declines on LastFM, and an obvious performance drop is observed on Yelp.

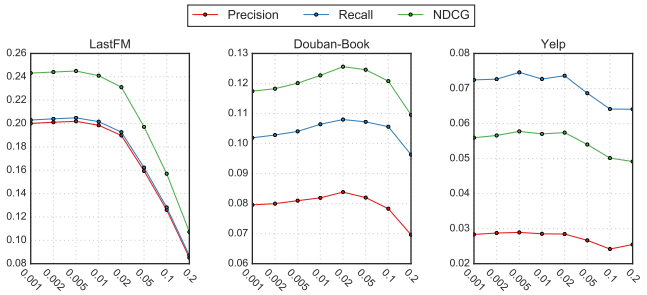

Figure 4: Sensitivity analysis of β.

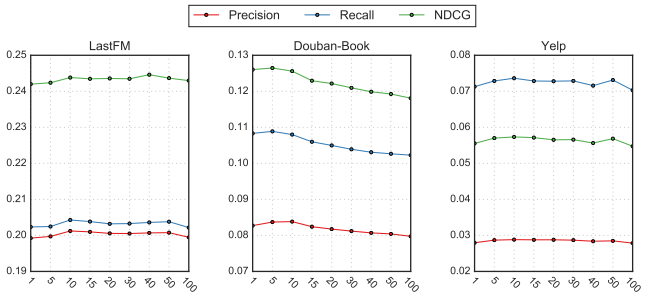

Figure 5: Influence of the number of used positive examples.

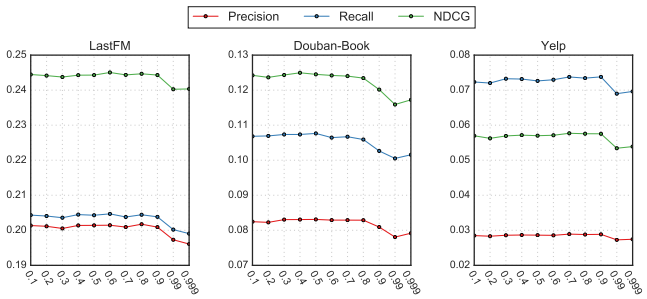

Figure 6: Influence of the edge dropout rate.

On Douban-Book, the performance firstly gets a slight rise and then declines obviously when there is only one view. The results demonstrate that, under the semi-supervised setting, even a single view can generate desirable self-supervised signals, which is encouraging since social relations or other side information are not always accessible in some situations. Besides, increasing the used number of views may bring more performance gains, but it is not absolutely right.

4.5 Parameter Sensitivity Analysis

There are three important hyper-parameters used in SEPT: β for controlling the magnitude of self-supervised tri-training, K - the number of used positive examples and ρ - the edge dropout rate of $\tilde{\mathcal{G}}$. We choose some representative values for them to investigate the parameter sensitivity of SEPT. The results are presented in Fig. 4 - 6. When investigating the influence of β, we fix $K = 10$ and $\rho = 0.3$. For the influence of K in Fig. 5, we fix $\beta = 0.005$ on LastFM and Yelp, $\beta = 0.02$ on Douban-Book, and $\rho = 0.3$. Finally, for the effect of ρ in Fig. 6, the setting of β is as the same as the last case, and $K = 10$. A two-layer setting is used in this case.

As can be observed from Fig. 4, SEPT is sensitive to β. On different datasets, we need to choose different values of β for the best performance. Generally, a small value of β can lead to a desirable performance, and a large value of β results in a huge performance drop. Figure 5 has been interpreted in Section 4.3. According to Fig. 6, we observe that SEPT is not sensitive to the edge dropout rate. Even a large value of ρ (e.g., 0.8) can create informative self-supervision signals, which is a good property for the possible wide use of SEPT. When the perturbed graph is highly sparse, it cannot provide useful information for self-supervised learning.

5 CONCLUSION AND FUTURE WORK

The self-supervised graph contrastive learning, which is widely used in the field of graph representation learning, recently has been transplanted to recommendation for improving the recommendation performance. However, most SSL-based methods only exploit self-supervision signals through the self-discrimination, and SSL cannot fully exert itself in the scenario of recommendation to leverage the widely observed homophily. To address this issue, in this paper, we propose a socially-aware self-supervised tri-training framework named SEPT to improve recommendation by discovering self-supervision signals from two complementary views of the raw data. Under the self-supervised tri-training scheme, the neighbor-discrimination based contrastive learning method is developed to refine user representations with pseudo-labels from the neighbors. Extensive experiments demonstrate the effectiveness of SEPT, and a thorough ablation study is conducted to verify the rationale of the self-supervised tri-training.

In this paper, only the self-supervision signals from users are exploited. However, items can also analogously provide informative pseudo-labels for self-supervision. This can be implemented by leveraging the multimodality of items. We leave it as our future work. We also believe that the idea of self-supervised multi-view co-training can be generalized to more scenarios beyond recommendation.

ACKNOWLEDGMENT

This work was supported by ARC Discovery Project (Grant No. DP190101985) and ARC Training Centre for Information Resilience (Grant No. IC200100022).

REFERENCES

[1] Philip Bachman, R Devon Hjelm, and William Buchwalter. 2019. Learning representations by maximizing mutual information across views. In *Advances in Neural Information Processing Systems*. 15535–15545.
[2] Rianne van den Berg, Thomas N Kipf, and Max Welling. 2017. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263* (2017).
[3] Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*. 92–100.
[4] Hongxu Chen, Hongzhi Yin, Tong Chen, Weiqing Wang, Xue Li, and Xia Hu. 2020. Social boosted recommendation with folded bipartite network embedding. *IEEE Transactions on Knowledge and Data Engineering* (2020).
[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*. PMLR, 1597–1607.

[6] Bradley Efron and Robert J Tibshirani. 1994. *An introduction to the bootstrap*. CRC press.

[7] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *arXiv preprint arXiv:1706.02216* (2017).

[8] Tengda Han, Weidi Xie, and Andrew Zisserman. 2020. Self-supervised co-training for video representation learning. *arXiv preprint arXiv:2010.09709* (2020).

[9] Kaveh Hassani and Amir Hosein Khasahmadi. 2020. Contrastive Multi-View Representation Learning on Graphs. *arXiv preprint arXiv:2006.05582* (2020).

[10] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9729–9738.

[11] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yong-Dong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. ACM, 639–648.

[12] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. 2018. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670* (2018).

[13] Hong Huang, Jie Tang, Sen Wu, Lu Liu, and Xiaoming Fu. 2014. Mining triadic closure patterns in social networks. In *Proceedings of the 23rd international conference on World wide web*. 499–504.

[14] Wei Jin, Tyler Derr, Haochen Liu, Yiqi Wang, Suhang Wang, Zitao Liu, and Jiliang Tang. 2020. Self-supervised learning on graphs: Deep insights and new direction. *arXiv preprint arXiv:2006.10141* (2020).

[15] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[16] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942* (2019).

[17] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Zhaoyu Wang, Li Mian, Jing Zhang, and Jie Tang. 2020. Self-supervised learning: Generative or contrastive. *arXiv preprint arXiv:2006.08218* 1, 2 (2020).

[18] Hao Ma, Dengyong Zhou, Chao Liu, Michael R Lyu, and Irwin King. 2011. Recommender systems with social regularization. In *Proceedings of the fourth ACM international conference on Web search and data mining*. ACM, 287–296.

[19] Jianxin Ma, Chang Zhou, Hongxia Yang, Peng Cui, Xin Wang, and Wenwu Zhu. 2020. Disentangled Self-Supervision in Sequential Recommenders. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 483–491.

[20] Miller Mcpherson, Lynn Smithlovin, and James M Cook. 2001. Birds of a feather: Homophily in social networks. *Annual Review of Sociology* 27, 1 (2001), 415–444.

[21] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* (2018).

[22] Zhen Peng, Wenbing Huang, Minnan Luo, Qinghua Zheng, Yu Rong, Tingyang Xu, and Junzhou Huang. 2020. Graph Representation Learning via Graphical Mutual Information Maximization. In *Proceedings of The Web Conference 2020*. 259–270.

[23] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. 2020. GCC: Graph Contrastive Coding for Graph Neural Network Pre-Training. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1150–1160.

[24] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, 452–461.

[25] Aravind Sankar, Yanhong Wu, Yuhang Wu, Wei Zhang, Hao Yang, and Hari Sundaram. 2020. GroupIM: A Mutual Information Maximization Framework for Neural Group Recommendation. *arXiv preprint arXiv:2006.03736* (2020).

[26] Ke Sun, Zhouchen Lin, and Zhanxing Zhu. 2019. Multi-Stage Self-Supervised Learning for Graph Convolutional Networks on Graphs with Few Labels. *arXiv preprint arXiv:1902.11038* (2019).

[27] Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. 2019. Deep Graph Infomax.. In *ICLR (Poster)*.

[28] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*. 165–174.

[29] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2020. Self-supervised Graph Learning for Recommendation. *arXiv preprint arXiv:2010.10783* (2020).

[30] Le Wu, Junwei Li, Peijie Sun, Yong Ge, and Meng Wang. 2020. DiffNet++: A Neural Influence and Interest Diffusion Network for Social Recommendation. *arXiv preprint arXiv:2002.00844* (2020).

[31] Le Wu, Peijie Sun, Yanjie Fu, Richang Hong, Xiting Wang, and Meng Wang. 2019. A Neural Influence Diffusion Model for Social Recommendation. *CoRR* abs/1904.10322 (2019).

[32] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 346–353.

[33] Shiwen Wu, Wentao Zhang, Fei Sun, and Bin Cui. 2020. Graph Neural Networks in Recommender Systems: A Survey. *arXiv preprint arXiv:2011.02260* (2020).

[34] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* (2020).

[35] Xin Xia, Hongzhi Yin, Junliang Yu, Qinyong Wang, Lizhen Cui, and Xiangliang Zhang. 2020. Self-Supervised Hypergraph Convolutional Networks for Session-based Recommendation. *arXiv preprint arXiv:2012.06852* (2020).

[36] Xin Xin, Alexandros Karatzoglou, Ioannis Arapakis, and Joemon M Jose. 2020. Self-Supervised Reinforcement Learning for Recommender Systems. *arXiv preprint arXiv:2006.05779* (2020).

[37] Tiansheng Yao, Xinyang Yi, Derek Zhiyuan Cheng, Felix Yu, Aditya Menon, Lichan Hong, Ed H Chi, Steve Tjoa, Evan Ettinger, et al. 2020. Self-supervised Learning for Deep Models in Recommendations. *arXiv preprint arXiv:2007.12865* (2020).

[38] Hongzhi Yin, Bin Cui, Ling Chen, Zhiting Hu, and Xiaofang Zhou. 2015. Dynamic user modeling in social media systems. *ACM Transactions on Information Systems (TOIS)* 33, 3 (2015), 1–44.

[39] Hongzhi Yin, Xiaofang Zhou, Bin Cui, Hao Wang, Kai Zheng, and Quoc Viet Hung Nguyen. 2016. Adapting to User Interest Drift for POI Recommendation. *IEEE Transactions on Knowledge and Data Engineering* 28, 10 (2016), 2566–2581.

[40] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph Contrastive Learning with Augmentations. *Advances in Neural Information Processing Systems* 33 (2020).

[41] Junliang Yu, Min Gao, Jundong Li, Hongzhi Yin, and Huan Liu. 2018. Adaptive Implicit Friends Identification over Heterogeneous Network for Social Recommendation. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. ACM, 357–366.

[42] Junliang Yu, Min Gao, Hongzhi Yin, Jundong Li, Chongming Gao, and Qinyong Wang. 2019. Generating reliable friends via adversarial training to improve social recommendation. In *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 768–777.

[43] Junliang Yu, Hongzhi Yin, Jundong Li, Min Gao, Zi Huang, and Lizhen Cui. 2020. Enhance Social Recommendation with Adversarial Graph Convolutional Networks. *arXiv preprint arXiv:2004.02340* (2020).

[44] Junliang Yu, Hongzhi Yin, Jundong Li, Qinyong Wang, Nguyen Quoc Viet Hung, and Xiangliang Zhang. 2021. Self-Supervised Multi-Channel Hypergraph Convolutional Network for Social Recommendation. *arXiv preprint arXiv:2101.06448* (2021).

[45] Xiaohua Zhai, Avital Oliver, Alexander Kolesnikov, and Lucas Beyer. 2019. S4l: Self-supervised semi-supervised learning. In *Proceedings of the IEEE international conference on computer vision*. 1476–1485.

[46] Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. 2020. Sˆ 3-Rec: Self-Supervised Learning for Sequential Recommendation with Mutual Information Maximization. *arXiv preprint arXiv:2008.07873* (2020).

[47] Zhi-Hua Zhou and Ming Li. 2005. Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Transactions on knowledge and Data Engineering* 17, 11 (2005), 1529–1541.