Revisiting Graph based Social Recommendation: A Distillation Enhanced Social Graph Network

Ye Tao
School of Software and Microelectronics, Peking University
Beijing, China
tao.ye@pku.edu.cn

Ying Li*
Key Lab of High Confidence Software Technologies (MOE) & National Engineering Center of Software Engineering, Peking University
Beijing, China
li.ying@pku.edu.cn

Su Zhang
Center for Data Science, Peking University
Beijing, China
zachzsu@pku.edu.cn

Zhirong Hou
ICBC Technology Co.,Ltd
Beijing, China
houzr@tech.icbc.com.cn

Zhonghai Wu
Key Lab of High Confidence Software Technologies (MOE) & National Engineering Center of Software Engineering, Peking University
Beijing, China
wuzh@pku.edu.cn

ABSTRACT

Social recommendation, which leverages social connections to construct Recommender Systems (RS), plays an important role in alleviating information overload. Recently, Graph Neural Networks (GNNs) have received increasing attention due to their great capacity for graph data. Since data in RS is essentially in the structure of graphs, GNN-based RS is flourishing. However, existing works lack in-depth thinking of social recommendations. These methods contain implicit assumptions that are not well analyzed in practice. To tackle these problems, we conduct statistical analyses on widely used social recommendation datasets. We design metrics to evaluate the social information, which can provide guidance about whether and how we should use this information in the RS task. Based on these analyses, we propose a Distillation Enhanced SocIal Graph Network (DESIGN). We train a model that integrates information from the user-item interaction graph and the user-user social graph and train two auxiliary models that only use one of the above graphs respectively. These models are trained simultaneously, where the knowledge distillation technique restricts the training process and makes them learn from each other. Our extensive experiments show that our model significantly and consistently outperforms the state-of-the-art competitors on real-world datasets.

CCS CONCEPTS

• **Information systems → Recommender systems**; • **Human-centered computing → Social recommendation**.

KEYWORDS

recommender systems, graph neural networks, social network

ACM Reference Format:
Ye Tao, Ying Li*, Su Zhang, Zhirong Hou, and Zhonghai Wu. 2022. Revisiting Graph based Social Recommendation: A Distillation Enhanced Social Graph Network. In *Proceedings of the ACM Web Conference 2022 (WWW '22), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3485447.3512003

1 INTRODUCTION

In the era of information explosion, Recommender Systems (RS) play an important role in alleviating information overload. Collaborative Filtering (CF) is the most widely used technique in RS. It learns user and item embeddings based on the historical user-item interactions, assuming that behaviorally similar users would exhibit similar preferences on items. Graph Neural Network (GNN) is the most popular among the various CF techniques. Recent years have witnessed the boom of GNNs, which are state-of-the-art methods on graph representation learning. Data in RS can be naturally represented as graphs, where users and items are denoted as nodes, and interactions such as clicks and purchases are denoted as edges. GNN-based recommender systems learn user and item embeddings by stacking multiple layers of graph convolution operations to iteratively propagate messages between nodes on the graph structure.

However, as most users only have limited behavior data, RS models that only rely on user-item historical interactions suffer from the data sparsity issue. Luckily, with the rapid development of social networks, more and more people like to build social relationships and share their preferences on these platforms. Recently, social recommendation has emerged as a promising direction and attracted increasing attention. As well supported by the social influence theory [4, 25], users usually acquire and disseminate information through those around them, implying that the underlying social connections of users can play a significant role in building an RS model. In the early stage, studies focus on social regularization

WWW '22, April 25–29, 2022, Virtual Event, Lyon, France.
© 2022 Association for Computing Machinery.
ACM ISBN 978-1-4503-9096-5/22/04…$15.00
https://doi.org/10.1145/3485447.3512003

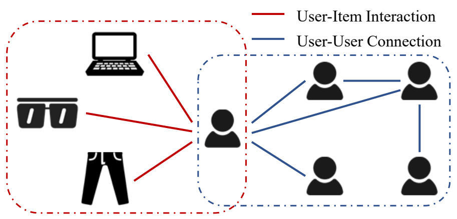

Figure 1: Graph data in the social recommender systems. It contains two graphs including the user-item bipartite graph (left part) and the user-user social graph (right part).

methods. Recently, GNN-based methods construct graph structured data to propagate messages between users and items, which help regularize the embedding and boost the predictive performance.

Nevertheless, we argue that existing GNN-based social recommendation methods lack in-depth thinking about their assumptions. From the graph signal processing perspective, several works argue that node embeddings consist of low-frequency true features and high-frequency noise [27, 37]. They have shown that GNN models iteratively aggregating neighborhood node embeddings corresponds to performing low-pass filtering on graph signals. Therefore, GNN models can learn better graph representations and improve performance because they perform low-pass filtering to denoise node features. It makes sense to perform such operations on the user-item interaction graph because the data to be predicted and historical interactions come from the same distribution. However, things are different for the social graph. Although the social influence theory [4, 25] states that socially connected people tend to have similar preferences, this tendency varies in different scenarios. When this tendency is not obvious, smoothing user embedding according to social connections will restrict the model's representational power and reduce its predictive performance.

To tackle the above issues, we conduct statistical analyses on four widely used social recommendation datasets. We design metrics to evaluate the tendency of socially connected users to have similar preferences. These metrics can reflect the significance of social connections for the corresponding recommendation task, providing guidance about whether and how to use social information for GNN-based recommendations. We experimentally prove that introducing low-quality social connections into the recommendation will hurt the predictive performance. To get a deeper understanding of the social influence theory [4, 25], we perform analyses to explore the interrelationship between (1) users interacting with items and (2) user building social connections with other users.

Based on the above analyses, we propose a Distillation Enhanced SocIal Graph Network (DESIGN). As shown in Fig 1, social recommendation datasets contain a user-item bipartite graph and a user-user social graph. Some RS methods only use one graph to perform convolution [3, 12, 34, 39], so they cannot well utilize all available information. Other works propose to integrate both graphs [7, 38]. However, these models tend to quickly over-fit the training data and cannot fully release their capabilities. To deal with these problems, we introduce the Knowledge Distillation (KD) technique into social recommendation. We propose to train a model that integrates knowledge from both graphs as the main model and train two models that only use one graph as auxiliary models respectively. Models that rely on the bipartite graph and the social graph perform different message propagation processes to smooth user representations. Their predictions have large differences, and each has its own advantages. All these models are trained simultaneously. The main model makes full use of all available knowledge to achieve better performance, and the auxiliary models restrict the training process of the main model, which can be seen as a regularization strategy.

To summarize, our main contributions are as follows. (1) We provide in-depth revisiting of the social influence theory by conducting statistical analyses on four widely used social recommendation datasets. We propose metrics to evaluate the social information, which can provide guidance about whether and how to leverage social connections among users to construct RS models. (2) We propose a Distillation Enhanced SocIal Graph Network (DESIGN) to improve the social recommendation predictive performance. To the best of our knowledge, we are the first to introduce the KD technique between models that rely on different data sources in the social recommendation task. (3) We perform extensive experiments on real-world datasets to show that the proposed DESIGN consistently outperforms the state-of-the-art competitors. **The data and source code underlying this article are released at** here.

2 RELATED WORK

Social Recommendation. In the early stage, studies mainly focused on MF-based methods. The leading philosophies can be categorized into three groups: SoRec [23] and TrustMF [45] are the representative works of co-factorization, STE [22] and mTrust [28] are typical ensemble methods, and SR [24] is a typical social regularization method. After the "renaissance" of neural networks, the boom of deep learning has broadened the ways to explore social recommendation [6, 8, 33, 36]. Some studies leverage deep structures to learn more complex representations. These models could be summarized into the following two categories: (i) social regularization based approaches [15, 40], which assume connected users would show similar embeddings; (ii) the user behavior enhancement based approaches [9, 10], which argued that social network provides valuable information to enhance each users' behavior.

GNN-based Recommendation. Recently, graph neural networks (GNNs) have received increasing attention due to their great capacity of learning on graph structure data [2, 5, 18, 26]. Early studies define the graph convolution operation on the spectral domain [5]. Later, GCN [18] and GraphSAGE [11] propose to define graph convolution in the spatial domain. Since most information in RS tasks essentially has graph structure, the field of GNN-based recommendation is flourishing [43]. Most GNN-based methods rely on the user-item interaction graph [1, 3, 12, 34, 35, 41, 46]. Recently, GNNs have also been employed for social recommendation. GraphRec [7] first introduces GNNs to this task by modeling user-user interactions as social graph data. Then, DiffNet [39] and its extension DiffNet++ [38] simulate how users are influenced by the recursive social diffusion process. DANSER [42] uses a dual graph attention network to collaboratively learn representations for two-fold social effects. ESRF [47] proposes a deep adversarial framework to address the common issues in social recommendation.

Table 1: Data statistics of social recommendation datasets.

	flickr	yelp	ciao	epinions
Users (U-I graph)	8252	17235	7375	22164
Users (U-U graph)	8358	17237	7375	22166
Items	82120	37378	105114	296277
Ratings	327815	207945	282650	912441
Rating density	0.048%	0.032%	0.036%	0.014%
Valid user pairs	4172140	6666124	5059691	32594272
Valid ratio	5.972%	2.244%	9.303%	6.634%
Social pairs	352952	259014	170410	574228
Valid social pairs	98468	82834	86004	54838
Valid social ratio	27.898%	31.981%	50.469%	9.549%
Social diffuse level	4.617	14.254	5.425	1.439
Social density	0.505%	0.087%	0.313%	0.117%
Valid social density	2.360%	1.243%	1.700%	0.168%

Knowledge Distillation. KD is a model-agnostic network compression technique to transfer the knowledge of a large model (teacher) to a small model (student) [14]. The transferred knowledge reveals hidden properties that are not explicitly included in the training set so that they accelerate and improve the learning of the student model, achieving a balance between effectiveness and efficiency. Some recent studies have employed KD for RS to reduce the size of models while maintaining the performance [16, 20, 21, 29]. However, they constructed different models by controlling the embedding size of users and items. By contrast, we build different GNN-based RS models that rely on different data sources.

3 PRELIMINARIES

3.1 Problem Definition

In the task of social recommendation, we denote $\mathcal{U} = \{u_1, u_2, ..., u_n\}$ as the user set and $\mathcal{V} = \{v_1, v_2, ..., v_m\}$ as the item set respectively, where n and m are the numbers of users and items. $\mathcal{G}_r = (\mathcal{U}, \mathcal{V}, \mathcal{E}_r)$ denotes the user-item bipartite graph where $(u, i) \in \mathcal{E}_r$ indicated that the user u purchased/rated the item i. $\mathcal{G}_s = (\mathcal{V}, \mathcal{E}_s)$ denotes the user-user social graph where $(u_1, u_2) \in \mathcal{E}_s$ indicated that user u_1 and u_2 are socially connected. Let user-item interaction matrix be $\mathbf{R} \in \mathbb{R}^{n \times m}$, where $\mathbf{R}_{ui} = 1$ if $(u, i) \in \mathcal{E}_r$ otherwise 0. The adjacency matrix of the user-item graph can be written as follows.

$$\mathbf{A} = \begin{pmatrix} \mathbf{0} & \mathbf{R} \\ \mathbf{R}^T & \mathbf{0} \end{pmatrix} \tag{1}$$

We denote the set of items which u have interacted with as $\mathcal{N}_r(u)$, the set of users who have interacted with i as $\mathcal{N}_r(i)$, and the set of users who have social connections with u as $\mathcal{N}_s(u)$. Given the available bipartite graph $\mathcal{G}_r$ and social graph $\mathcal{G}_s$, social recommendation models aim to predict the missing value in $\mathcal{G}_r$.

3.2 Data Analyses

We perform statistical analyses on four widely used recommendation datasets that contain social information. *Yelp* is an online location-based social network. *Flickr* is a who-trust-whom online image-based social sharing platform. *Ciao* and *Epinions* are taken from popular social networking websites Ciao (www.ciao.co.uk) and Epinions (www.epinions.com) that allow users to rate items, browse/write reviews, and add friends. We perform statistic analyses to verify the social influence theory [4, 25], i.e., whether socially connected users tend to have similar preferences on the above datasets. Specifically, we care about three research questions:

- RQ1: Compared with two random users, whether socially connected users have a higher probability of having similar preferences? If so, how much is the probability difference?
- RQ2: What is the probability that two users with similar preferences have a social relationship? Whether the amount of information provided by social relationships is significant?
- RQ3: What kind of users pairs are more likely to be socially connected. Does it have a similar trend with the message propagation process of GNN-based RS models?

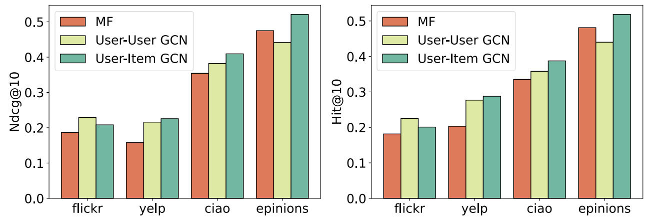

Figure 2: Experimental results of three representative RS models on four widely used social recommendation datasets.

In the recommendation task, the user-item interactions are the best evidence to evaluate whether users have similar preferences. To simplify the illustration, we define a function val to evaluate whether two users have a similar preference. We define (u_1, u_2) as a *valid* user pair if $\exists v_i, v_i \in \mathcal{N}_r(u_1) \cap \mathcal{N}_r(u_2)$ (two users interact with at least one same item), and $val(u_1, u_2) = 1$. Otherwise $val(u_1, u_2) = 0$. All data statistics are summarized in table 1. For all experiments, we randomly divide the data into the training/validation/test set with the proportion of 80%/ 5%/15% by default.

3.2.1 ***RQ1:*** In a bipartite graph $\mathcal{G}_r$ with n users, there are $n \times n$ candidate user pairs. We calculate (1) the ratio of *valid* user pairs among all candidates, (2) the ratio of *valid* user pairs among all socially connected user pairs, and (3) the ratio between the above two metrics. These three metrics are formally defined as follows:

$$\text{Valid ratio} = \frac{\sum_{u_i} \sum_{u_j} val(u_i, u_j)}{n \times n} \tag{2}$$

$$\text{Valid social ratio} = \frac{\sum_{(u_i, u_j) \in \mathcal{G}_s} val(u_i, u_j)}{\sum_{(u_i, u_j) \in \mathcal{G}_s}} \tag{3}$$

$$\text{Social diffuse level} = \frac{\text{Valid social ratio}}{\text{Valid ratio}} \tag{4}$$

Generally speaking, if the *social diffuse level* in a dataset is high (significantly larger than 1.0), we can conclude that socially connected users do have a higher probability to have similar preferences.

To illustrate the relevance of these metrics and social recommendation tasks, we conducted experiments and compared the results among the following representative methods: (i) MF [19] that directly projects the single ID of a user or an item into an

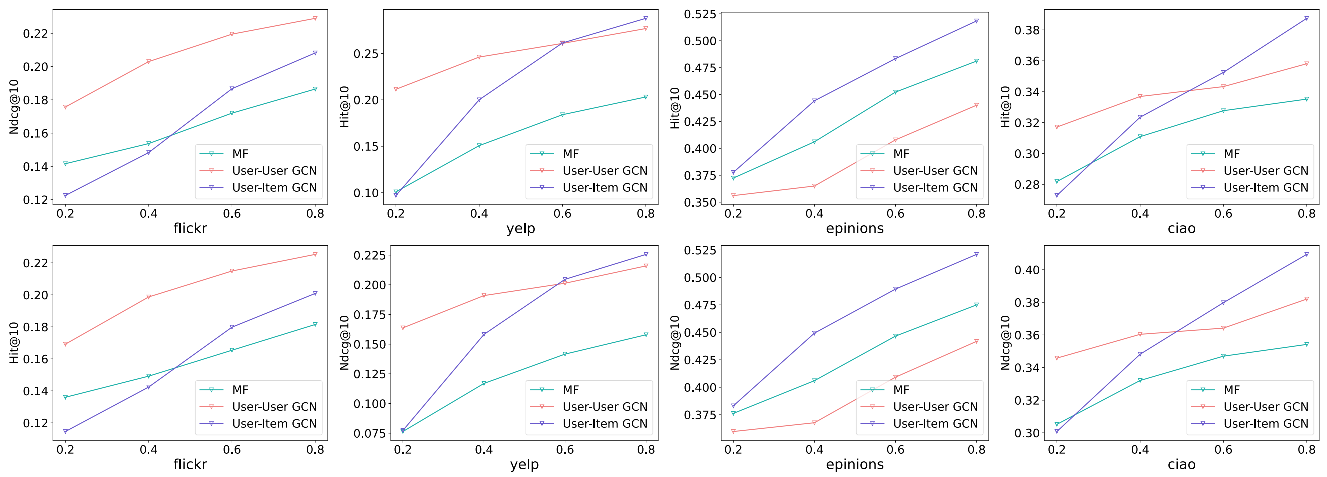

Figure 3: Experimental results on four social recommendation datasets with different proportions of training data.

embedding vector, (ii) User-User GCN that performs graph convolution on the social graph to simulate the recursive social diffusion (which corresponds to DiffNet [39]), and (iii) User-Item GCN that performs graph convolution operation on the bipartite graph to exploit the high-order connectivity from history interactions (which corresponds to LightGCN [12]). The results are shown in Fig 2.

User-Item GCN outperforms MF on all datasets, proving that performing node embedding smoothing with graph convolution on the user-item graph always denoises node features and enhances the predictive performance. By contrast, User-User GCN only outperforms MF on three datasets. On epinions, smoothing the user embedding with the social graph actually hurts the performance. In fact, we can expect this result without training the model. The *social diffuse level* is low in epinions, i.e., only slightly larger than 1.0, meaning that the probability that two users with social relationships have similar preferences is not significantly higher than that of two random users. Therefore, restricting socially connected users to have similar embeddings with GNN will limit the model's expressive power. It is worth noting that this result is due to the inherent properties of GNN models. We have also tried to use more advanced GNN structures for social diffusion on epinions (such as GAT and GraphSAGE) and came to the same conclusion.

3.2.2 ***RQ2:*** The social graph enhances the recommender system by providing extra information. We assume that social connections will become critical for training a robust RS model when the user-item interaction graph is sparse and the user-user social graph contains high-quality information. We calculate the ratio of *valid* user pairs that are socially connected among all *valid* user pairs:

$$\text{Valid social density} = \frac{\sum_{(u_i,u_j)\in\mathcal{G}_s} val(u_i,u_j)}{\sum_{u_i}\sum_{u_j} val(u_i,u_j)} \tag{5}$$

Generally, if the *valid social density* is high (the social graph provides rich information) and the rating density is low (the user-item interaction is sparse), the information provided by the social relationships will be critical for building a robust RS model. To verify our conclusion, we conducted experiments with different amounts of training data. Since *valid social ratio* is much larger than *valid social density* in all datasets, using less training data leads to the increase of *valid social density* and the decrease of *rating density*.

The results in Fig 3 show that User-Item GCN performs poorly when there are few training data (such as only 20% data for training). As the amount of training data increases, its performance increases rapidly, which has obvious advantages over the MF model. This result shows that only when there is sufficient training data, a satisfactory result can be obtained by performing graph convolution on the user-item graph to smooth the node embedding. By contrast, User-User GCN always outperforms MF on datasets with high *social diffuse level* regardless of the training data proportion. Except for epinions, on datasets that social connections can well reflect user preference' similarities, User-User GCN outperforms the other two models with a big gap when the training data is sparse. User-Item GCN can achieve better performance on *yelp* and *ciao* with sufficient training data. Therefore, we conclude that: (1) When the training data is sparse, high-quality social information is essential for training a recommendation model, (2) When there is sufficient training data, using the user-item bipartite graph to smooth the user node embeddings usually obtains better performance.

3.2.3 ***RQ3:*** Performing graph convolution on the bipartite graph $\mathcal{G}_r$ or the social graph $\mathcal{G}_s$ both propagate messages between users. There are two most widely used graph convolution operations on the user-item bipartite graph: (i) standard GCN with symmetric normalization and (ii) GCN with a mean aggregator. Let the k-th layer embedding matrix be $\mathbf{E}^k \in \mathbb{R}^{(n+m)\times d}$, the above two operations are

$$\text{Standard GCN:} \qquad \mathbf{E}^{k+1} = \left(\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}\right)\mathbf{E}^k \tag{6}$$

$$\text{Mean GCN:} \qquad \mathbf{E}^{k+1} = \left(\mathbf{A}\mathbf{D}^{-1}\right)\mathbf{E}^k \tag{7}$$

where $\mathbf{D} \in \mathbb{R}^{(m+n)\times(m+n)}$ is the degree matrix, in which each entry $\mathbf{D}_{ii}$ denotes the number of nonzero entries in the i-th row vector of the adjacency matrix $\mathbf{A}$. If we concentrate on the user representations and analyze a 2-layer model, the graph convolution operation on $\mathcal{G}_r$ can be regarded as performing embedding smoothing over

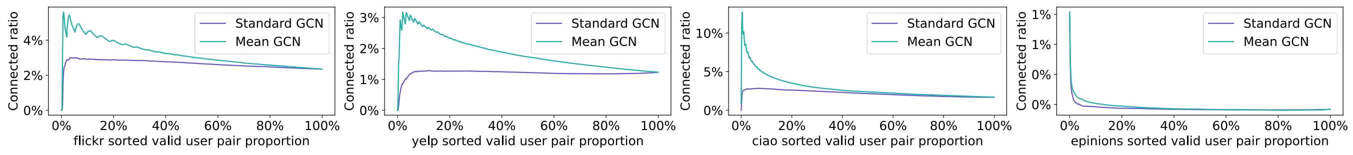

Figure 4: Statistics of social connected ratio between the user pairs sorted by the propagation weights corresponding to the two graph convolution operations. We only sort and plot *valid* user pairs that have interact with same items.

user pairs that have overlap on the interacted items. More specifically, we denote $p_u^{k+2} = \sum_v c_{v->u} p_v^k$, where $c_{v->u}$ are called propagation weights of valid user pair (v, u). The $c_{v->u}$ for the above two widely used graph convolution operations are as follows

$$c_{v->u} = \frac{1}{\sqrt{|\mathcal{N}_r(u)|}\sqrt{|\mathcal{N}_r(v)|}} \sum_{i \in \mathcal{N}_r(u) \cap \mathcal{N}_r(v)} \frac{1}{|\mathcal{N}_r(i)|} \tag{8}$$

$$c_{v->u} = \frac{1}{|\mathcal{N}_r(u)|} \sum_{i \in \mathcal{N}_r(u) \cap \mathcal{N}_r(v)} \frac{1}{|\mathcal{N}_r(i)|} \tag{9}$$

We sorted all *valid* user pairs according to these propagation weights, i.e., (u_i, v_i) being ahead of (u_j, v_j) in the sorted *valid* user pairs list means $c_{u_i->v_i} \geq c_{u_j->v_j}$. Then, we plot the ratio of *valid* user pairs that are socially connected in Fig 4, i.e., a data point (10%, 4%) means that there are 4% user pairs that are socially connected among the top 10% valid user pairs (sorted by propagation weights).

The result is a little surprising. Models employ the standard GCN interpret the propagation weights from v to u as follows: (1) the number of co-interacted items, the more the larger; (2) the popularity of the co-interacted items, the less popularity the larger; and (3) the activity of v and u, the less active the larger. However, we observe the user pairs that exchange more information in the standard GCN model do not have a higher probability of being socially connected compared with a random *valid* user pair (ratio curves are flat from 0% to 100%). By contrast, the socially connected probabilities do have a similar trend with the message propagation process of mean GCN. This illustrates that the first two assumptions are reasonable, while the third should be adjusted to *Active users are less affected by each item they interact with, but their influence on other users will not be reduced by their increased activity.*

4 DISTILLATION ENHANCED SOCIAL RECOMMENDATION SYSTEM

This section first proposes a Distillation Enhanced SocIal Graph Network (DESIGN) framework to deal with the social recommendation tasks. Then, we provide implementation details for each component in our DESIGN framework. Lastly, we describe how to perform model prediction training for our DESIGN.

4.1 DESIGN Framework

The basic idea of GNN-based RS models is to smooth node features over the graph. As illustrated in previous research, GNNs iteratively aggregating neighborhood node embeddings correspond to performing low-pass filtering on graph signals [27, 37]. Therefore, the key point is to choose a suitable filter. In social recommendation, we have a user-item graph $\mathcal{G}_r$ and a user-user graph $\mathcal{G}_s$. It is natural to construct the filter in two ways: (1) Building a filter according to a single graph or (2) combining the two graphs to build a unified filter. However, both methods have limitations. For the first method, only using one graph cannot well utilize all available information. For the second method, directly combining the two graph structures without imposing restrictions on the model will quickly result in over-fitting and cannot fully release its capabilities. To solve these problems, we propose a new framework to comprehensively utilize information from various data sources.

According to the results of RQ3, although social relationships can reflect users' preference similarity, there are lots of differences between the filters constructed according to $\mathcal{G}_r$ and $\mathcal{G}_s$. Using these two filters separately to smooth the node embedding will result in two models that have different characteristics. We propose to employ the Knowledge Distillation (KD) technique to take advantage of this feature to improve their performances further. KD is a model-agnostic strategy that trains a model with the guidance of other models [14, 20]. We claim that models relying on $\mathcal{G}_r$ or $\mathcal{G}_s$ have their own advantages. Performing KD can empower them to take advantage of each other's complementary knowledge. In order to make full use of all available data, we propose to train a model that integrates $\mathcal{G}_r$ and $\mathcal{G}_s$ as the main model and train two auxiliary models that rely on $\mathcal{G}_r$ or $\mathcal{G}_s$ respectively. These three models are defined as TwinGCN, RatingGCN, and SocialGCN. Fig 5 shows an overview of our DESIGN framework. All models are trained simultaneously by using each other's knowledge along with the binary labels. The training strategy are as follows:

$$\mathcal{L}_t(\theta_t) = \mathcal{L}_{CF}(\theta_t) + \lambda_{r\to t} \cdot \mathcal{L}_{KD}(\theta_t; \theta_r) + \lambda_{s\to t} \cdot \mathcal{L}_{KD}(\theta_t; \theta_s) \tag{10}$$

$$\mathcal{L}_r(\theta_r) = \mathcal{L}_{CF}(\theta_r) + \lambda_{t\to r} \cdot \mathcal{L}_{KD}(\theta_r; \theta_t) + \lambda_{s\to r} \cdot \mathcal{L}_{KD}(\theta_r; \theta_s) \tag{11}$$

$$\mathcal{L}_s(\theta_s) = \mathcal{L}_{CF}(\theta_s) + \lambda_{t\to s} \cdot \mathcal{L}_{KD}(\theta_s; \theta_t) + \lambda_{r\to s} \cdot \mathcal{L}_{KD}(\theta_s; \theta_r) \tag{12}$$

$$\mathcal{L}_{DESIGN} = \mathcal{L}_t + \mathcal{L}_r + \mathcal{L}_s \tag{13}$$

where t, r and s denote TwinGCN, RatingGCN and SocialGCN respectively, θ_* is the model parameters. $\mathcal{L}_{CF}$ is the collaborative filtering loss depending on each single model, $\mathcal{L}_{KD}$ is the bidirectional distillation loss and $\lambda_{a\to b}$ are the hyper-parameters that control the effects of the distillation loss from model a to b. In our experiment, we simply set the $\lambda_{*\to *}$ to 1 without further tuning.

4.2 GNN-based Social Recommendation

The core of GNN models is how to aggregate messages from neighbor nodes. Such operation can be abstracted as:

$$h_u^{k+1} = \text{Combine}\left(h_u^k, \text{Aggregate}\left(\left\{h_v^k : v \in \mathcal{N}_u\right\}\right)\right) \tag{14}$$

Many works have specified the *Aggregate* function, such as the sum, mean, max, and attention-based aggregators [11, 31, 44]. As for GNN-based RS models, most works tie embedding affine transformations and nonlinear activation with the *Combine* function.

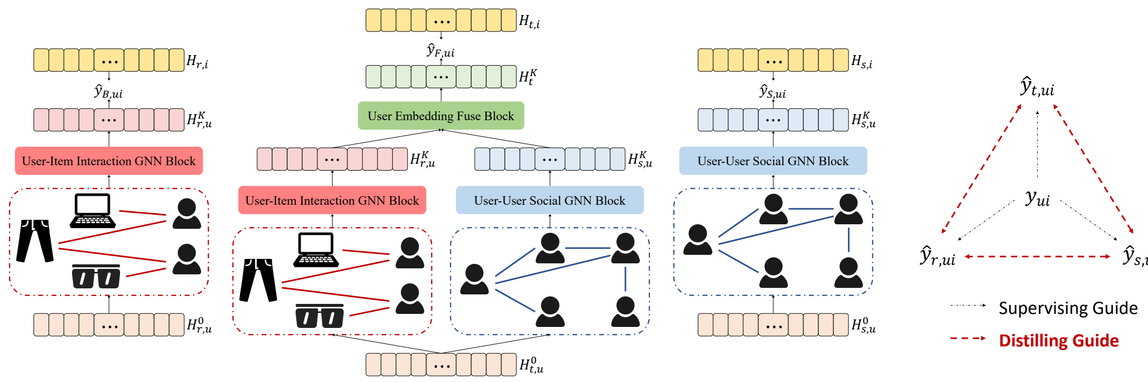

Figure 5: Overview of our Distillation Enhanced SocIal Graph Network (DESIGN) framework.

However, some recent studies have illustrated that these complex operations could be burdensome for collaborative filtering while a simple propagation could achieve better performance [3, 12]. Following their design philosophy, we propose to use simple structures to implement our GNN-based RS models.

We first specify the implementation for the SocialGCN as

$$h^{k+1}_{s,u} = \sum_{v \in \mathcal{N}_s(u)} \frac{1}{|\mathcal{N}_s(u)|} h^k_{s,v} \tag{15}$$

where $|\mathcal{N}_s(u)|$ is the number of elements in $\mathcal{N}_s(u)$. Then, we specify the implementation for the RatingGCN. Most existing GNN-based RS models on $\mathcal{G}_r$ smooth both the user and item embeddings. However, we argue that performing representation smoothing over item nodes will hardly improve the predictive performance and sometimes even limit the expressive power and reduce the performance. Although there are certain similarities between multiple items purchased by the same user, the use of GNN for smoothing will impose too strong restrictions on item embeddings. Therefore, we only perform embedding smoothing for user nodes on $\mathcal{G}_r$.

$$h^{k+1}_{r,u} = \sum_{i \in \mathcal{N}_r(u)} \frac{1}{|\mathcal{N}_r(u)|} \sum_{v \in \mathcal{N}_r(i)} \frac{1}{|\mathcal{N}_r(i)|} h^k_{s,v} \tag{16}$$

where $|\mathcal{N}_r(i)|$ and $|\mathcal{N}_r(u)|$ are the numbers of elements in $\mathcal{N}_r(u)$ and $\mathcal{N}_s(u)$. **According to the result of RQ3** in Section 3.2, we employ the mean aggregator instead of the standard GCN with symmetric normalization to perform graph convolution on $\mathcal{G}_r$. Finally, we specify the implementation for the TwinGCN. **According to the result of RQ2** in Section 3.2, GNN-based RS models that rely on $\mathcal{G}_s$ and $\mathcal{G}_r$ each have their own advantages under different circumstances. To train a robust model that performs well in various scenarios, we combine these models as follows.

$$h^k_{t,u} = \alpha_{r,u} \cdot h^{k_1}_{r,u} + \alpha_{s,u} \cdot h^{k_2}_{s,u} \tag{17}$$

where $\alpha_{r,u}$ and $\alpha_{s,u}$ are the weights of user embedding from $\mathcal{G}_r$ and $\mathcal{G}_s$, which can be treated as a hyper-parameter to be tuned manually, or as a model parameter (e.g., output of an attention network) to be optimized automatically. In our experiments, we find that setting $\alpha_{r,u} = \alpha_{s,u} = \frac{1}{2}$ leads to good performance. Thus we do not tune this hyper-parameter to keep its simplicity. We use SocialGCN and RatingGCN as sub-structures of TwinGCN and let them share the free embedding before the graph convolution operation, i.e., $h^0_{r,u} = h^0_{s,u}$. Since the two sub-structures can be regarded as smoothing the user embedding with different filters, we simply use the weighted sum to fuse their results. Besides, because the message aggregation process of the two sub-structure does not interfere with each other, we can fuse their representations from different layers. In our experiments, we just set $k = k_1 = k_2$ and leave the relevant discussions as future work.

4.3 Model Prediction

In our proposed DESIGN framework, the only trainable parameters are $\{H_{t,i}, H_{r,i}, H_{s,i}\}$ for all items and $\{H_{t,u}, H_{r,u}, H_{s,u}\}$ for all users. When these parameters are given, user embeddings at higher layers can be computed according to Equation 15-17. After K layers graph convolution operation, we further combine these embeddings obtained at each layer to form the final users' representations.

$$h_{r,u} = \sum_{k=1}^{K} \alpha_{r,k} h^k_{r,u};\ h_{s,u} = \sum_{k=1}^{K} \alpha_{s,k} h^k_{s,u};\ h_{t,u} = \sum_{k=1}^{K} \alpha_{t,k} h^k_{t,u} \tag{18}$$

where $\alpha_{*,k} \geq 0$ denotes the importance of the k-th layer in constituting the final embedding, which can be tuned manually or treated as a parameter. In our experiment, we just set $\alpha_{*,k}$ uniformly as $1/K$ to keep its simplicity. All models get prediction by performing inner product of users' and items' final representations:

$$\hat{y}_{r,ui} = h^T_{r,u} h_{r,i}; \quad \hat{y}_{s,ui} = h^T_{s,u} h_{s,i}; \quad \hat{y}_{t,ui} = h^T_{t,u} h_{t,i} \tag{19}$$

where $\hat{y}_{*,ui}$ is the ranking score for recommendation generation.

4.4 Model Training

To train SocialGCN, RatingGCN, and TwinGCN under the framework according to Equation 10-13, we need to specify the CF loss and the KD loss. As we focus on implicit feedbacks of users, we follow [13] to regard the recommendation task as a binary classification task. We set the target value y_{ui} to a binarized 1 or 0 denoting whether u has interacted with i. And we employ a probabilistic approach for optimization, where we need to restrict the prediction score in the range $[0, 1]$ to represent how likely i is relevant to u. With the above settings, we define the CF loss as follows.

$$L_{CF}(\theta_*) = -\sum_{(u,i)} y_{ui} \ln \sigma(\hat{y}_{*,ui}) + (1 - y_{ui}) \ln(1 - \sigma(\hat{y}_{*,ui})) \tag{20}$$

Table 2: Comparison between DESIGN and baseline methods. All improvements are statistically significant for p < 0.05.

Dataset	Training ratio	Metric	MF	GraphRec	NGCF	LR-GCCF	DiffNet	DiffNet++	LightGCN	DESIGN
flickr	80%	HR@10	0.1815	0.1676	0.1649	0.2019	0.2253	0.1650	0.2009	**0.2517**
		NDCG@10	0.1865	0.1704	0.1705	0.2057	0.2290	0.1702	0.2082	**0.2590**
	20%	HR@10	0.1360	0.1395	0.1373	0.1091	0.1692	0.1383	0.1146	**0.1719**
		NDCG@10	0.1416	0.1446	0.1457	0.1171	0.1757	0.1440	0.1225	**0.1809**
yelp	80%	HR@10	0.2031	0.2170	0.2194	0.2700	0.2768	0.2131	0.2878	**0.2944**
		NDCG@10	0.1579	0.1693	0.1723	0.2106	0.2159	0.1674	0.2206	**0.2266**
	20%	HR@10	0.1009	0.0911	0.0909	0.0919	0.2114	0.0908	0.0970	**0.2168**
		NDCG@10	0.0764	0.0709	0.0709	0.0747	0.1637	0.0697	0.0773	**0.1713**
ciao	80%	HR@10	0.3352	0.3327	0.3372	0.3854	0.3542	0.3381	0.3875	**0.4262**
		NDCG@10	0.3542	0.3514	0.3559	0.4069	0.3820	0.3579	0.4094	**0.4483**
	20%	HR@10	0.2818	0.2860	0.2845	0.2747	0.3171	0.2815	0.2728	**0.3314**
		NDCG@10	0.3052	0.3062	0.3055	0.2989	0.3458	0.3045	0.3008	**0.3576**

Since we regard the recommendation task as a binary classification task, we use the cross-entropy loss to distill knowledge from each other. Specifically, we define the KD loss as follows.

$$L_{KD}(\theta_S;\theta_T) = -\sum_{(u,i)} \sigma(\hat{y}_{T,ui}) \ln \sigma(\hat{y}_{S,ui}) - (1-\hat{y}_{T,ui}) \ln(1-\sigma(\hat{y}_{S,ui})) \tag{21}$$

where S and T denote the student model and teacher model respectively. For each KD loss function, we cut off the gradient of the teacher model and only use it to train the student model.

As we could only observe positive feedbacks with huge missing unobserved values, we randomly sample 8 unobserved feedbacks for each positive feedback as pseudo negative samples at each iteration. The advanced sampling strategies are postponed as future work.

5 EXPERIMENT

5.1 Experimental Settings

Dataset. According to the result of RQ1 in Section 3.2, the social information in *flickr*, *yelp* and *ciao* are helpful to the GNN-based recommendation while the social information in *epinions* are not well aligned with user's preference similarity and may hurt the predictive performance. Since we mainly concentrate on the GNN-based social recommendation task, we only conduct experiments on the first three datasets. The data statistics are listed in Table 1.

Baseline Methods. We choose competitive baselines, including MF [19], GrapgRec [7] that combines representations from the user-item and user-user graph, NGCF [34] that employs standard GCN on the user-user graph, LR-GCCF [3] that linearly propagates embedding on the user-item graph, DiffNet [39]/DiffNet++ [38] that perform embedding diffusion on the user-user graph and LightGCN [12] that simplify NGCF by removing feature transformation.

Parameter Settings. For all GNN-based methods, we search the number of layers in the range of $\{2, 3, 4\}$ and implement them with Deep Graph Library (DGL) [32]. The embedding size is fixed to 64 for all models. We optimize all models using Adam [17] optimizer with the learning rate 0.001, where the batch size is fixed to 512.

5.2 Comparison with State-of-the-Art Methods

Table 2 shows the performance comparison between our model and state-of-the-art competitors on HR@10 and NDCG@10. We report the result of TwinGCN under the DESIGN framework as the performance of our model, which consistently and significantly outperforms all baseline methods on all datasets. Although some models perform well with sufficient training data, their performances are not satisfactory with sparse training data. By contrast, our model performs well under all circumstances. The results demonstrate the effectiveness of the knowledge distillation technique and our simple but elegant model structure. It is worth noting that DESIGN can be further improved by tuning its hyper-parameters, such as the $\lambda_{*\to *}$ in Equation 10-12 and the $\alpha_{*,u}$ in Equation 17.

Moreover, we observed methods that employ complex model structures, including GraphRec, NGCF, and DiffNet++, have poor performance due to severe over-fitting issues. These models assume there are raw user/item features. However, since some social recommendation datasets do not have this information, we learn user/item representations from free embeddings. Under this setting, models that remove the redundant complex operations achieve betters performance. Based on the experimental results, we conclude that GNN-based RS models should concentrate on message propagation with simple structures. If there exist raw node features of users or items, we can design a special structure to process it before graph convolution operation, as one does in [30].

Table 3: Performance comparison of HR@10 and NDCG@10 between DESIGN and its alternative implementation choices.

Model	Flickr		Yelp		Ciao	
	HR	NDCG	HR	NDCG	HR	NDCG
DESIGN	0.2517	0.2590	0.2944	0.2266	0.4262	0.4483
Dim-distill	0.2352	0.2407	0.2802	0.2143	0.4050	0.4236
Transform	0.2192	0.2235	0.2758	0.2137	0.3966	0.4186
Sym-norm	0.2469	0.2532	0.2846	0.2189	0.4097	0.4334
Item-smooth	0.2470	0.2530	0.2905	0.2232	0.4123	0.4347

5.3 Design Choice Analyses

To verify the superiority of our design choice, we provide a comparison of the proposed model and alternative design choices, including (1) Instead of the DESIGN framework, we construct two TwinGCN models with different dimensions and employ the bi-directional KD technique to train these models like [20], which is denoted as *Dim-distill*; (2) Adding additional feature transformation and nonlinear activation to update the target node representations (i.e., from GCN to GraphSAGE), which is denoted as *Extra-transform*; (3) Using symmetric normalization to aggregate neighbors' messages, which is denoted as *Sym-norm*; (4) Not only smoothing the user embeddings but also smoothing the item embeddings in RatingGCN and TwinGCN, which is denoted as *Item-smooth*.

The results in Table 3 illustrate the effectiveness of our choices. Constructing different models with different sizes by changing the embedding dimensions is the dominant strategy in the current KD technique for RS models. However, the result of *Dim-distill* shows that although it can slightly improve the performance, our DESIGN framework is a better way to enhance the social recommendation models with the KD technique. The result of *Transform* proves that our simple structure can outperform more complex ones that tend to over-fit. **The result of *Sym-norm* verifies our conclusion of RQ3**, i.e., mean GCN is better than symmetric-norm for aggregating messages on $\mathcal{G}_r$. The result of *Item-smooth* shows performing graph convolution to smooth item embedding increases computational load without improving performance. When there are more items and fewer users, such as *ciao*, it even hurts the performance.

5.4 Knowledge Distillation Analyses

To further illustrate the effectiveness of our KD technique design, we try to train models under our DESIGN framework separately without introducing the KD loss, which are denoted *-single respectively. Then we add the KD loss and jointly train these models. The results are presented in Fig 6. We have the following observations:

(1) Introducing the KD technique consistently yields significant improvement to the three models. When using 80% data for training, the distilled models have the average performance improvement *w.r.t* HR@10 by 7.09%, 6.08%, 5.74% and NDCG@10 by 6.91%, 6.99%, 5.30% in *flickr*, *yelp*, *ciao* respectively compared with single-models. SocialGCN, RatingGCN, and TwinGCN can capture different patterns that are helpful for the downstream tasks. Therefore, introducing the KD loss to let them learn from each other can effectively optimize the training process and alleviate the over-fitting problem.

(2) SocialGCN and RatingGCN each have their own advantages. SocialGCN outperforms RatingGCN when the data sparsity problem is serious. As the amount of training data increases, RatingGCN not only has more training data but also optimizes its user embedding smoothing process. Therefore, its performance increases faster, gradually catching up or even outperforming the performance of SocialGCN. **These results are in line with the conclusion of RQ2** in Section 3.2. TwinGCN integrates the user embedding in RatingGCN and SocialGCN and inherits the advantages of both methods. This integration enables TwinGCN to be well adapted to various situations, yielding a more robust and better performance.

5.5 Hyper-parameter Analyses

To provide a more comprehensive understanding of the DESIGN framework, we examine the effects of important hyper-parameters $\lambda_{*\to*}$ that controls the KD losses. We set all $\lambda_{*\to*}$ to the same value from $\{0.0, 0.1, 1.0, 10.0\}$. The results in Fig 7 show that the best performances are achieved when $\lambda_{*\to*} = 1$ across datasets. The performance increases first and then decreases as the constraints

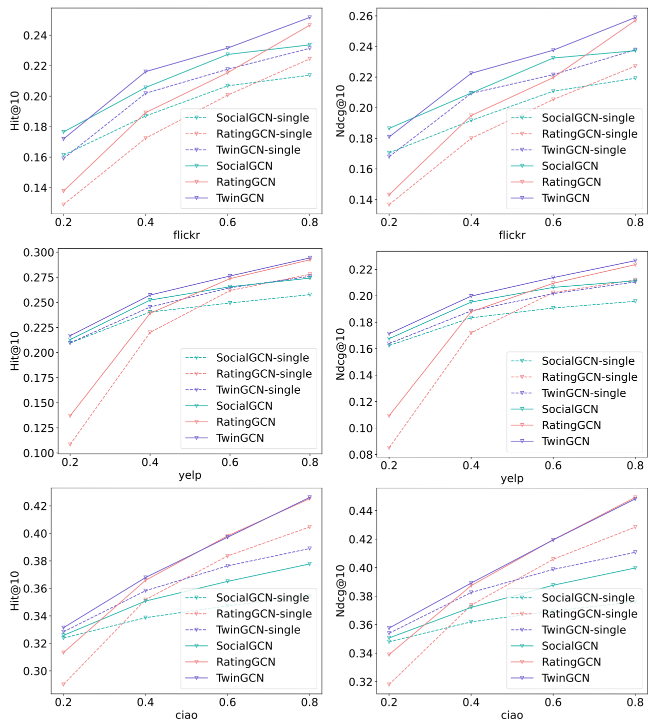

Figure 6: Performance comparison between SocialGCN, RatingGCN, and TwinGCN. *-single denotes the result of training a single model separately without introducing the KD loss to learn from other models' predictions.

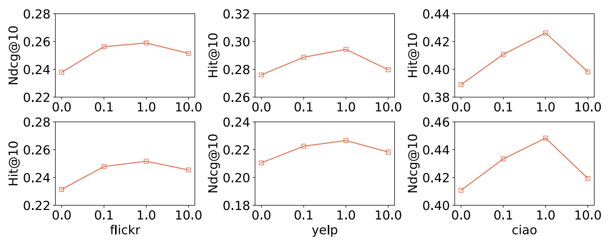

Figure 7: Performance comparison of with various $\lambda_{*\to*}$ that controls the balance between the CF loss and the KD loss.

increase. When $\lambda_{*\to*}$ is too small, the training process of each model is not sufficiently regularized. Therefore, they still tend to over-fit the training data. When the constraints are too strong, the capabilities of the main model will be limited by the models with insufficient capabilities, resulting in performance degradation.

6 CONCLUSION

In this paper, we perform statistical data analyses to obtain a deeper understanding of the social influence theory. Based on the analyses, we propose a Distillation Enhanced SocIal Graph Network (DESIGN). Under the DESIGN framework, we introduce the knowledge distillation between models that rely on different data sources to leverage the social information for the recommendation task. Our extensive experiments on real-world datasets show that our model outperforms the state-of-the-art competitors.

ACKNOWLEDGMENTS

This work is partly supported by ICBC technology.

REFERENCES

[1] Rianne van den Berg, Thomas N Kipf, and Max Welling. 2017. Graph convolutional matrix completion. *arXiv preprint arXiv:1706.02263* (2017).

[2] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. 2017. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine* 34, 4 (2017), 18–42.

[3] Lei Chen, Le Wu, Richang Hong, Kun Zhang, and Meng Wang. 2020. Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 27–34.

[4] Robert B Cialdini and Noah J Goldstein. 2004. Social influence: Compliance and conformity. *Annu. Rev. Psychol.* 55 (2004), 591–621.

[5] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems* 29 (2016), 3844–3852.

[6] Wenqi Fan, Qing Li, and Min Cheng. 2018. Deep modeling of social relations for recommendation. In *Thirty-Second AAAI Conference on Artificial Intelligence.*

[7] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *The World Wide Web Conference.* 417–426.

[8] Wenqi Fan, Yao Ma, Dawei Yin, Jianping Wang, Jiliang Tang, and Qing Li. 2019. Deep social collaborative filtering. In *Proceedings of the 13th ACM Conference on Recommender Systems.* 305–313.

[9] Guibing Guo, Jie Zhang, and Neil Yorke-Smith. 2015. Trustsvd: Collaborative filtering with both the explicit and implicit influence of user trust and of item ratings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 29.

[10] Guibing Guo, Jie Zhang, and Neil Yorke-Smith. 2016. A novel recommendation model regularized with user trust and item ratings. *ieee transactions on knowledge and data engineering* 28, 7 (2016), 1607–1620.

[11] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems.* 1025–1035.

[12] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong 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.* 639–648.

[13] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web.* 173–182.

[14] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[15] Meng Jiang, Peng Cui, Fei Wang, Wenwu Zhu, and Shiqiang Yang. 2014. Scalable recommendation with social contextual information. *IEEE Transactions on Knowledge and Data Engineering* 26, 11 (2014), 2789–2802.

[16] SeongKu Kang, Junyoung Hwang, Wonbin Kweon, and Hwanjo Yu. 2020. DE-RRD: A Knowledge Distillation Framework for Recommender System. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management.* 605–614.

[17] Diederik P Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *arXiv: Learning* (2014).

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

[19] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.

[20] Wonbin Kweon, SeongKu Kang, and Hwanjo Yu. 2021. Bidirectional Distillation for Top-K Recommender System. In *Proceedings of the Web Conference 2021.* 3861–3871.

[21] Jae-woong Lee, Minjin Choi, Jongwuk Lee, and Hyunjung Shim. 2019. Collaborative distillation for top-N recommendation. In *2019 IEEE International Conference on Data Mining (ICDM).* IEEE, 369–378.

[22] Hao Ma, Irwin King, and Michael R Lyu. 2009. Learning to recommend with social trust ensemble. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval.* 203–210.

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

[24] 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.* 287–296.

[25] Miller McPherson, Lynn Smith-Lovin, and James M Cook. 2001. Birds of a feather: Homophily in social networks. *Annual review of sociology* 27, 1 (2001), 415–444.

[26] Andriy Mnih and Russ R Salakhutdinov. 2008. Probabilistic matrix factorization. In *Advances in neural information processing systems.* 1257–1264.

[27] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of graph neural network evaluation. *arXiv preprint arXiv:1811.05868* (2018).

[28] Jiliang Tang, Huiji Gao, and Huan Liu. 2012. mTrust: Discerning multi-faceted trust in a connected world. In *Proceedings of the fifth ACM international conference on Web search and data mining.* 93–102.

[29] Jiaxi Tang and Ke Wang. 2018. Ranking distillation: Learning compact ranking models with high performance for recommender system. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* 2289–2298.

[30] Ye Tao, Ying Li, and Zhonghai Wu. 2021. Revisiting Graph Neural Networks for Node Classification in Heterogeneous Graphs. In *2021 IEEE International Conference on Multimedia and Expo (ICME).* IEEE, 1–6.

[31] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[32] Minjie Wang, Lingfan Yu, Da Zheng, Quan Gan, Yu Gai, Zihao Ye, Mufei Li, Jinjing Zhou, Qi Huang, Chao Ma, et al. 2019. Deep Graph Library: Towards Efficient and Scalable Deep Learning on Graphs. (2019).

[33] Xiang Wang, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. 2017. Item silk road: Recommending items from information domains to social users. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval.* 185–194.

[34] 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.

[35] Xiang Wang, Hongye Jin, An Zhang, Xiangnan He, Tong Xu, and Tat-Seng Chua. 2020. Disentangled graph collaborative filtering. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval.* 1001–1010.

[36] Yufei Wen, Lei Guo, Zhumin Chen, and Jun Ma. 2018. Network embedding based recommendation method in social networks. In *Companion Proceedings of the The Web Conference 2018.* 11–12.

[37] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *International conference on machine learning.* PMLR, 6861–6871.

[38] Le Wu, Junwei Li, Peijie Sun, Richang Hong, Yong Ge, and Meng Wang. 2020. Diffnet++: A neural influence and interest diffusion network for social recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2020).

[39] Le Wu, Peijie Sun, Yanjie Fu, Richang Hong, Xiting Wang, and Meng Wang. 2019. A neural influence diffusion model for social recommendation. In *Proceedings of the 42nd international ACM SIGIR conference on research and development in information retrieval.* 235–244.

[40] Le Wu, Peijie Sun, Richang Hong, Yong Ge, and Meng Wang. 2018. Collaborative neural social recommendation. *IEEE transactions on systems, man, and cybernetics: systems* (2018).

[41] Le Wu, Yonghui Yang, Kun Zhang, Richang Hong, Yanjie Fu, and Meng Wang. 2020. Joint item recommendation and attribute inference: An adaptive graph convolutional network approach. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval.* 679–688.

[42] Qitian Wu, Hengrui Zhang, Xiaofeng Gao, Peng He, Paul Weng, Han Gao, and Guihai Chen. 2019. Dual graph attention networks for deep latent representation of multifaceted social effects in recommender systems. In *The World Wide Web Conference.* 2091–2102.

[43] Shiwen Wu, Fei Sun, Wentao Zhang, and Bin Cui. 2020. Graph neural networks in recommender systems: a survey. *arXiv preprint arXiv:2011.02260* (2020).

[44] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* (2018).

[45] Bo Yang, Yu Lei, Jiming Liu, and Wenjie Li. 2016. Social collaborative filtering by trust. *IEEE transactions on pattern analysis and machine intelligence* 39, 8 (2016), 1633–1647.

[46] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* 974–983.

[47] Junliang Yu, Hongzhi Yin, Jundong Li, Min Gao, Zi Huang, and Lizhen Cui. 2020. Enhance social recommendation with adversarial graph convolutional networks. *IEEE Transactions on Knowledge and Data Engineering* (2020).