Federated Social Recommendation with Graph Neural Network

ZHIWEI LIU, LIANGWEI YANG, and ZIWEI FAN, University of Illinois at Chicago
HAO PENG, Beihang University
PHILIP S. YU, University of Illinois at Chicago

Recommender systems have become prosperous nowadays, designed to predict users' potential interests in items by learning embeddings. Recent developments of the Graph Neural Networks (GNNs) also provide recommender systems (RSs) with powerful backbones to learn embeddings from a user-item graph. However, only leveraging the user-item interactions suffers from the cold-start issue due to the difficulty in data collection. Hence, current endeavors propose fusing social information with user-item interactions to alleviate it, which is the social recommendation problem. Existing work employs GNNs to aggregate both social links and user-item interactions simultaneously. However, they all require centralized storage of the social links and item interactions of users, which leads to privacy concerns. Additionally, according to strict privacy protection under General Data Protection Regulation, centralized data storage may not be feasible in the future, urging a decentralized framework of social recommendation.

As a result, we design a federated learning recommender system for the social recommendation task, which is rather challenging because of its heterogeneity, personalization, and privacy protection requirements. To this end, we devise a novel framework **Fe**drated **So**cial recommendation with **G**raph neural network (FeSoG). Firstly, FeSoG adopts relational attention and aggregation to handle heterogeneity. Secondly, FeSoG infers user embeddings using local data to retain personalization. Last but not least, the proposed model employs pseudo-labeling techniques with item sampling to protect the privacy and enhance training. Extensive experiments on three real-world datasets justify the effectiveness of FeSoG in completing social recommendation and privacy protection. We are the first work proposing a federated learning framework for social recommendation to the best of our knowledge.

CCS Concepts: • **Information systems** → **Information retrieval**; • **Security and privacy**; • **Computing methodologies** → *Machine learning*;

Additional Key Words and Phrases: Federated learning, recommender system, social recommendation, graph neural network

Hao Peng is supported by the National Key R&D Program of China through grant 2021YFB1714800, NSFC through grants 62002007 and U20B2053, S&T Program of Hebei through grant 21340301D, Fundamental Research Funds for the Central Universities. Philip S. Yu is partially supported by NSF under grants III-1763325, III-1909323, III-2106758, and SaTC-1930941.
Authors' addresses: Z. Liu, L. Yang, Z. Fan, and P. S. Yu, Department of Computer Science, University of Illinois at Chicago, Chicago, IL 60607; emails: {zliu213, lyang84, zfan20, psyu}@uic.edu; H. Peng (corresponding author), School of Cyber Science and Technology, Beihang University, No. 37 Xue Yuan Road, Haidian District, Beijing, 100191, China; email: penghao@act.buaa.edu.cn.

2157-6904/2022/08-ART55 $15.00
https://doi.org/10.1145/3501815

ACM Reference format:
Zhiwei Liu, Liangwei Yang, Ziwei Fan, Hao Peng, and Philip S. Yu. 2022. Federated Social Recommendation with Graph Neural Network. *ACM Trans. Intell. Syst. Technol.* 13, 4, Article 55 (August 2022), 24 pages.
https://doi.org/10.1145/3501815

1 INTRODUCTION

The developments of **Recommender Systems** (**RSs**) [9, 28, 30, 40, 65] become prosperous nowadays. A well-designed recommender system is able to predict users' potential interests in items. The core of it is to learn user/item embeddings [29, 40, 66] by fitting historical user-item interactions. Recently, the prosperity of **graph neural networks** (**GNNs**) [3, 6, 27, 38] provide powerful frameworks to learn node embeddings, which also motivates the community to design GNN-based RS models [9, 29, 30, 50, 51]. However, the cold-start issue [28, 66], which is associated with users having few records, impairs the performance of learned embeddings.

To cope with this, one can leverage the social information of users [9, 25, 36, 42]. In this way, we assume that users with social links also share similar item interests. Therefore, we could simultaneously aggregate social information and user-item interactions [9, 10, 56, 61] to alleviate the cold-start issue. SocialGCN [55, 56] employs the **Graph Convolutional Network** (**GCN**) to enhance user embedding by simulating how the recursive social diffusion process influences users. GraphRec [9] and GraphRec+ [10] propose to model three types of aggregations upon social graph, user-item graph, and item-item graph. Thus, it can comprehensively fuse the social links and item transactions. ConsisRec [61] introduces the social inconsistency problem from context-level and relation-level. It solves this problem by using a sampling-based attention mechanism.

Though being effective in fusing the social and user-item information, they all require a centralized storage [5, 53, 62] of both the social networks and item transaction history of users. Existing centralized storage methods pose risks of leaking privacy-sensitive data. Additionally, due to the strict privacy protection under **General Data Protection Regulation** (**GDPR**),[1] centralized data storage may not be the first choice of online platforms in the future. Therefore, a new decentralized model training framework for social recommendation is necessary. According to previous researches in federated learning [34, 62], the user data can be stored locally in each client while only uploading the necessary gradients for updating the model on a server. As for the federated recommender systems [1, 4, 5, 53], the sensitive user-item interactions are stored locally and clients only upload gradients to update user/item embeddings.

However, there is few work discussing how to design a federated learning recommender system to complete the social recommendation task. We address the challenges of a **federated social recommender system** (**FSRS**) as follows: (1) *Heterogeneity*. The current federated recommender system stores user-item interactions locally. However, an FSRS requires both user-user and user-item interactions, as shown in Figure 1. Therefore, we should store and fuse two types of relations simultaneously. (2) *Personalization*. Each client has special item interests and social connections, which leads to the non-iid distribution of the local data [62]. The model should be able to characterize the personalized federated learning process [8] for those clients, which is rather challenging. (3) *Privacy Protection*. Though user privacy data are stored locally, a federated recommender system yet demands collecting necessary gradients [4] from clients for updating embeddings on the server. This uploading process may lead to information leakage of original data [4]. Therefore, we should design a protection module before uploading any information.

To this end, we devise a novel FSRS framework to address the challenges as mentioned above, which are **Fedrated Social recommendation with Graph neural network** (**FeSoG**). Firstly,

[1] https://gdpr-info.eu/.

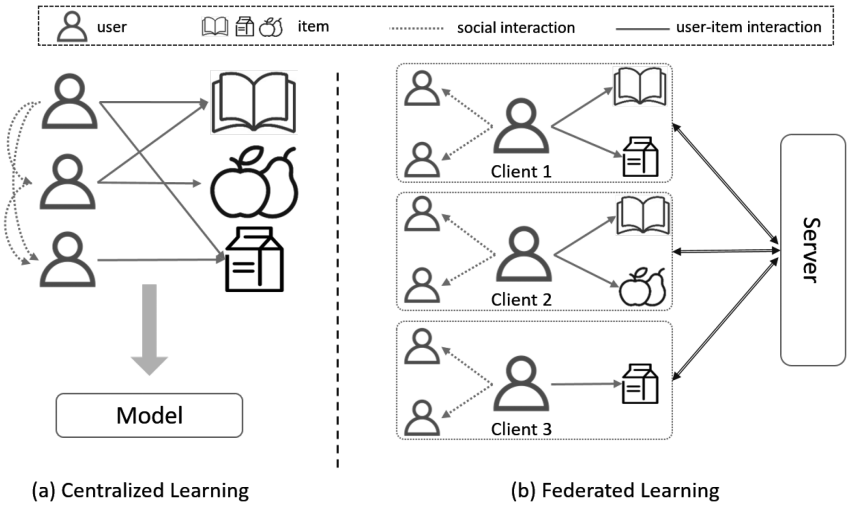


Fig. 1. Centralized learning (a) and federated learning (b) for social recommendation. The centralized learning trains the model with all the user privacy data, i.e., both social interactions and user-item interactions, available on the server. In contrast, federated learning locally stores user privacy data, only uploading and requesting non-sensitive data from the server. Dash lines between users denotes social interactions, while solid lines between users and items denotes user-item interactions.

we propose to use a local GNN to learn node embeddings. To tackle the heterogeneity of local data, we employ a relation attention mechanism to distinguish user-user and user-item interactions, which characterize the importance of neighbors by assigning different attention weights with respect to their relations. Secondly, the local GNNs on each client are updated only based on the data on devices. As such, mo*a*dels on devices possess personalizing training. Last but not least, FeSoG employs pseudo-labeling technique for the on-device training of local models. This pseudo-labeling can protect the privacy data from leakage when uploading gradients and enhance the robustness of the training process. Extensive experiments on three datasets demonstrate the effectiveness of FeSoG. Compared with other baselines, FeSoG achieves up to 5.26% in RMSE and 5.46% in MAE across all three datasets. In ablation studies, FeSoG also demonstrates the necessity of each component developed in the federated learning framework. The contributions are summarized as follows:

— **Novel:** We are the first work proposing a federated learning framework to tackle the social recommendation problem to the best of our knowledge.
— **Substantial:** We address three critical challenges, i.e., heterogeneity, personalization, and privacy protection, by proposing a new model FeSoG.
— **Comprehensive:** We conduct extensive experiments on three publicly available datasets to verify the effectiveness of FeSoG. Detailed analysis and ablation study further prove the efficacy of our proposed components in FeSoG.

In the following sections, we first introduce the related work in Section 2. Then, we present some preliminaries in Section 3, including both the definition and formulation. The detailed descriptions of our proposed FeSoG model are in Section 4. Experiments are discussed in Section 5. Finally, we conclude this article and open up possible future work in Section 6.

2 RELATED WORK

This section presents three relevant areas to this article: GNN for recommendation, social recommendation, and federated learning for recommendation.

2.1 Graph Neural Network for Recommendation

The recent developments of **Graph Neural Networks (GNNs)** [15, 21, 27, 49] motivate the community to propose a GNN-based recommender system. The intuition of a GNN model is to aggregate neighbors to recursively learn node embeddings [37]. GC-MC [2] first employs the GCN [21] architecture to complete the user-item rating matrix. It uses the GCN as an encoder to train user/item embeddings, which are input to a fully connected neural network to predict the ratings. PinSAGE [63] proposes to use the GraphSAGE [15] backbone to learn item embeddings over an attributed item graph. It first samples fixed-size nodes from multi-hop neighbors and then uses aggregators to aggregate those sampled nodes to learn the embeddings for center nodes. NGCF [51] is proposed later to explicitly model the collaborative signals upon user-item interaction graph by applying the GNN model. DGCF [29] observes the oscillation problem when applying GNN on the bipartite graph and solves it with cross-hop propagation layers. BasConv [30] is a pioneer work that investigates using GNN to complete basket recommendation. These works prove the efficacy of using the GNN framework to learn embeddings in a recommender system. GNN-based models are advantageous as their aggregation can model high-order structural information crucial for learning user/item embeddings from interactions. This article also adopts the GNN model to embed the local graphs. We employ the graph attention networks [48] as a backbone.

2.2 Social Recommendation

The social recommendation aims at relieving the data sparsity and cold start problem by inducing information of social links between users [32, 52, 61]. Social recommendation methods can be generally categorized as social **matrix factorization** (**MF**)-based methods and GNN- based methods. Existing social MF approaches either jointly factorize the rating and social relationship matrices or regularize the user/item embeddings with constraints of social connections. SoRec [32] co-factorizes the user rating matrix and social link matrix. SocialMF [18] adds a regularization term to constrain the difference between the user's taste and his/her trusted friends' average weighted taste. SoReg [33] adds a regularization term to directly minimize the difference in the user latent feature between two trusted users, which can prevent the counteraction of the latent feature of one's trusted friends. HGMF [52] introduces a **hierarchical group matrix factorization** (**HGMF**) technique to learn the user-group feature in a social network for recommendation. Unlike MF methods, GNN methods infer node embeddings directly from graphs and demonstrate the effectiveness from recent social recommendation work [24, 57, 58]. GraphRec [9] and GraphRec+ [10] use graph attention networks to learn user and item embeddings for recommendation. Reference [44] utilizes dynamic graph attention networks to capture the dynamic user's interest from the social dimension. CUNE [64] assumes that users hold implicit social links from each other. CUNE extracts semantic and reliable social information by graph embedding method. DiffNet [55] and DiffNet++ [54] model the social influence diffusion process to enhance the social recommendation. ConsisRec [61] examines the inconsistency problems in the social recommendation and introduces a consistent neighbor sampling module in the GNN model. The above studies show the effectiveness of incorporating social information into the recommender system.

2.3 Federated Learning for Recommender System

Google proposed Federated learning in 2016 [34]. It calls for data privacy-preserving solutions in machine learning models [7], with the raised privacy concerns of existing centralized

training-based models. The fundamental of federated learning is to design a decentralized training framework, which distributes the data to clients rather than storing it in a server [22, 34]. User transactions are sensitive information and probably cause identity information leakage if used for malicious purposes. Several recent works [1, 4, 53] developed federated recommender systems for user information protection while still preserving good enough personalization. **Federated Collaborative Filtering** (**FCF**) [1] and FedMF [4] are two pioneering works investigating a novel federated learning framework to learn the user/item embeddings for a recommender system. Both works develop the federated learning on the top of factorization [23] of the user-item rating matrix. To achieve federated learning, they propose that the user's ratings should be stored locally. The user embeddings can be trained locally, and the server only retains the item embeddings. This training framework leads to protecting the privacy data, as there is no transfer of users' interactions. Ribero et al. [41] argues that the model updates sent to the server may contain sufficient information to uncover raw data, which leaves privacy concerns. They propose to use the differential privacy [35] to limit the exposure of the data in a federated recommender system. FED-MVMF [11] extends the **Multi-View Matrix Factorization** (**MVMF**) [43] to a federated learning framework. It simultaneously factorizes both feature matrices and interaction matrices. A-FRS [5] proposes a robust federated recommender system against the poisoning attacks of clients. It employs an item similarity model [19] in learning the user/item embeddings. FedGNN [53] is the most recent work that combines GNN with a federated recommender system, which is also the most relevant work to our article. However, FedGNN fails to solve social recommendations, and the clients' models are not personalized [8]. We present a comparison of a set of representative social recommender systems and federated learning methods in Table 1.

3 PRELIMINARY

In this section, we present the preliminaries and definitions of essential concepts. The glossary of necessary notations are summarized in Table 2.

3.1 Definitions

The target in a social recommendation is to predict the users' ratings to items, when given social interactions and user-item interactions. Denote the $\mathcal{U} = \{u_1, u_2, \ldots, u_N\}$ and $\mathcal{T} = \{t_1, t_2, \ldots, t_M\}$ as the set of users and item, respectively. N and M are the numbers of users and items, respectively. The social recommendation is to complete the ratings of users to items given both rating matrix $\mathbf{R} \in \mathbb{R}^{N \times M}$ and the social connection matrix $\mathbf{S} \in \{0, 1\}^{N \times N}$. We denote the user n's rating value to an item m as $\mathbf{R}_{nm}$. Similarly, the connection between an user n and user p is denoted as $\mathbf{S}_{np}$. In a federated learning scenario, the data of each user are stored locally. Hence, both the rating matrix and social connection matrix are not available. The data of each user are stored in the local client, which is defined as

Definition 1 (Client). A client c is defined as a local device storing the rating data and the social data. Each client c_n is associated with a user n, whose rating data and social data are $\mathbf{R}_{n\cdot}$ and $\mathbf{S}_{n\cdot}$, respectively.

Definition 2 (Server). A server is defined as a central device managing the coordination of multiple clients in training a model. It does not exchange raw data from clients but only requests necessary messages for updating the model.

In this article, we assume clients and server to be honest-but-curious [20]. In other words, they provide correct information and cannot tamper with the training process. The **FSRS** is to complete the rating matrix given its partially complete rating data and social data, which is defined as follows:

Table 1. Comparison of Representative Models with Respect to Social Information, Multi-relation, GNN, Rating Protection, Interaction Protection, and Data Storage

	Social Information	Multi-relation	Graph Neural Network	Data Storage
RSTE [31]	✓	×	×	centralized
TrustWalker [17]	✓	×	×	centralized
SoRec [32]	✓	×	×	centralized
SoReg [33]	✓	×	×	centralized
SocialMF [18]	✓	×	×	centralized
TrustSVD [14]	✓	×	×	centralized
CUNE [64]	✓	✓	×	centralized
GCMC+SN [2]	✓	×	✓	centralized
DANSER [58]	✓	✓	✓	centralized
GraphRec [9]	✓	✓	✓	centralized
ConsisRec [61]	✓	✓	✓	centralized
FedMF [4]	×	×	×	local
FedGNN [53]	×	×	✓	local
FeSoG	✓	✓	✓	local

Definition 3 (FSRS). For n clients and a server, given the partially observed rating data and social data as $\mathbf{r}_n = [r_{n1}, r_{n2}, \ldots, r_{nk}]$ and $\mathbf{s}_n = [s_{n1}, s_{n2}, \ldots, s_{np}]$ of each client c_n, respectively, where $n, p \in \{1, 2, \ldots, N\}$ and $k \in \{1, 2, \ldots, M\}$, an FSRS can predict the unobserved rating data of the client c_n without access to the raw data in each client.

Note that both the rating and social data are stored locally in the corresponding clients and never be uploaded to the server. Multiple clients collaboratively train an FSRS under the orchestration of the center server [20].

3.2 Formulation

Our FSRS is designed by formulating the data on clients as multiple local graphs, which is illustrated in Figure 1. The local graph contains the first order neighbors of the client user, including item ratings and social neighbors. We denote the local graph for client c_n as $\mathcal{G}_n$, which consists of both user nodes and item nodes. $\mathcal{G}_n$ is constructed from partially observed privacy data. Moreover, there are two type of edges in $\mathcal{G}_n$, i.e., the user-item edges with rating value as attributes and the user-user edges denoting the social interactions. For each client c_n, we denotes its rated items as $\mathcal{T}^{(n)} = \{t_1^{(n)}, t_2^{(n)}, \ldots, t_k^{(n)}\}$ and social neighbors as $\mathcal{U}^{(n)} = \{u_1^{(n)}, u_2^{(n)}, \ldots, u_p^{(n)}\}$. The FSRS can predict the rating value of an unobserved item $t^* \in \mathcal{T} \setminus \mathcal{T}^{(n)}$. In other words, the FSRS can predict the attribute value of the local graph for the edge between the user u_i and a new item t^*. Thus, the problem can be formulated as follows:

Definition 4 (Problem Definition). Given the local graphs $\{\mathcal{G}_n|_{n=1}^N\}$, can we collaboratively train a model to predict the attribute value for an unobserved edges (u_n, t^*) without access to the raw data of any local graphs?

We specify the social recommendation problem to be a link prediction problem. It indicates that we should learn graph embeddings from local graphs to preserve the structural information. Additionally, it is necessary to tackle the heterogeneity [16, 30, 60] of those local graphs.

Table 2. Glossary of Notations

Symbol	Definition
$\mathcal{U}; \mathcal{T}$	user set; item set
$\mathbf{R}; \mathbf{S}$	rating matrix; social connection matrix
$N; M$	total number of users; total number of items
c_i	client n, which is associated with user n
$\mathbf{r}_n; \mathbf{s}_n$	the local observed rating and social data of client c_n
$\mathcal{T}^{(n)}; \mathcal{U}^{(n)}$	the local observed rated items; the social connected neighbors of client c_n
$\mathbf{E}_u(\mathbf{e}_u); \mathbf{E}_t(\mathbf{e}_t)$	embedding for users, embedding for items
$\mathbf{e}^*_{u_n}$	the local inference embedding of user n
$\mathbf{a}; \mathbf{b}; \mathbf{c}$	attention layer vector for user-user interaction; for item-item interaction; for relation vector
$\mathbf{W}_1; \mathbf{W}_2$	linear mapping matrix for user-user interaction; for item-item interaction
α_{up}, β_{uk}	attention weights for neighbor users; for neighbor items
$\mathbf{h}_u; \mathbf{h}_t$	hidden embeddings for neighbor users; hidden embeddings for neighbor items
$\gamma_u; \gamma_t$	attention weight for aggregating social relation; for aggregating user-item relation
$\mathbf{v}_u; \mathbf{v}_t$	social relation vector; user-item relation vector
$\mathbf{R}; \hat{\mathbf{R}}$	ground truth rating score; predicted rating score
$\mathcal{L}_u; \tilde{\mathcal{L}}_u$	loss for client u; protected loss for client u
$\mathbf{g}^{(n)}; \mathbf{g}^{(n)}_e; \mathbf{g}^{(n)}_m$	the gradients for client n; the embedding gradients; the model gradients
$\delta; \lambda$	the parameters for LDP

4 PROPOSED FRAMEWORK

In this section, we illustrate the FSRS framework of our proposed FeSoG. It has three crucial modules: embeddings layer, local GNNs, and gradient protector. The proposed framework is in Figure 3.

4.1 Embeddings

Node embeddings are crucial components in preserving graph structural information [30, 51, 61]. The FeSoG has embedding layers for user and item nodes. We denote the embeddings for users and items as $\mathbf{E}_u \in \mathbb{R}^{d \times N}$ and $\mathbf{E}_t \in \mathbb{R}^{d \times N}$, respectively, which are both maintained by the server. $d \in \mathbb{N}_+$ is the dimension size for embeddings. Clients request the embeddings tables from the server. Then, they learn a local user/item embeddings and a local GNN model by using their interaction data. Those embeddings will be updated on the server by aggregating the gradients uploaded from clients.

A client downloads the complete embedding tables and uses the user/item ids in interaction records to infer the corresponding embeddings. To be more specific, for a client n, which has rated items as $\mathcal{T}^{(n)} = \{t^{(n)}_k|^K_{k=1}\}$ and social neighbors as $\mathcal{U}^{(n)} = \{u^{(n)}_p|^P_{p=1}\}$, its rated item embeddings are $\{\mathbf{e}^{(n)}_{t_k}|^K_{k=1}\}$ and social neighbor embeddings are $\{\mathbf{e}^{(n)}_{u_p}|^P_{p=1}\}$, where $\mathbf{e}^{(n)}_{t_k}, \mathbf{e}^{(n)}_{u_p} \in \mathbb{R}^d$ and K, P denote the

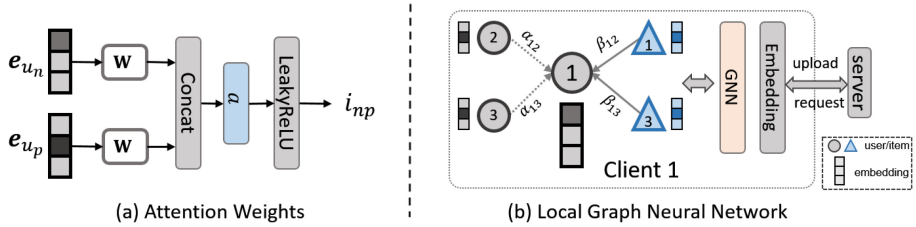


Fig. 2. (a): The calculation of the attention weights between two embeddings, which consists of linear mapping matrix $\mathbf{W}$, concatenation of two embeddings, attention layer a, non-linear activation, and softmax function. (b): The local GNN learns the embedding of u_1 by aggregating the neighboring embeddings.

total number of item neighbors and user neighbors, respectively. Those embeddings are input the local GNN model to learn client user embedding and predict item scores.

4.2 Local Graph Neural Network

The local GNN module is the major component in FeSoG to learn node embeddings and make a prediction. It consists of heterogeneous graph attention layers, relational graph aggregation layers, and rating prediction layers.

4.2.1 Relational Graph Attention. In general, we have no constraints on the local GNNs. There can be arbitrary GNN models, such as GCN [21], GAT [49], and GraphSAGE [15], and so on. This article focuses on proposing a new FSRS framework. We directly adopt the GAT layer for learning node embedding and leave other GNN layers for future investigation. The GAT layer is designed by employing the self-attention mechanism [48]. To learn the node embedding of u_n, we aggregate neighbor embeddings of u_n. However, since neighbors contribute unequally to the center node u_n, we should first learn a weight for each neighbor by employing an attention layer. Specifically, for a social pair (u_n, u_p), their attention scores are formulated as

$$o_{np} = \text{Attention}(\mathbf{W}_1\mathbf{e}_{u_n}, \mathbf{W}_1\mathbf{e}_{u_p}), \tag{1}$$

where o_{np} is a scalar denoting the attention weight, $\mathbf{W}_1 \in \mathbb{R}^{d\times d}$ is a linear mapping matrix, and Attention is the attention layer. We define the attention layer as a single-layer feed-forward neural network. It is parameterized with a weight vector $a \in \mathbb{R}^{2d}$ and employs a LeakyReLU activation [48], which is formulated as

$$o_{np} = \text{LeakyReLU}\left(\mathbf{a}^\top \left[\mathbf{W}_1\mathbf{e}_{u_n} \| \mathbf{W}_1\mathbf{e}_{u_p}\right]\right), \tag{2}$$

where $\mathbf{a}^\top$ denotes the transpose of the attention layer parameter and $\|$ denotes the concatenation operation of two vectors. An illustration of the attention weight is in Figure 2(a). The attention weight should be calculated for all the neighbors of the center node u, which forms a probability distribution by employing the softmax function:

$$\alpha_{np} = \text{softmax}_p(o_{np}) = \frac{\exp(o_{np})}{\sum_{i=1}^{P}\exp(o_{ni})}, \tag{3}$$

where α_{np} is the final attention weights, exp denotes the exponential function. Note that α is calculated as the attention weights for user neighbors. We should learn attention weights for user-item pair (u_n, i_k) in a similar way, which employs another linear mapping as in Equation (1) and another attention parameters as in Equation (4), as following:

$$v_{nk} = \text{LeakyReLU}\left(\mathbf{b}^\top \left[\mathbf{W}_2\mathbf{e}_{u_n} \| \mathbf{W}_2\mathbf{e}_{i_k}\right]\right), \tag{4}$$

where $\mathbf{W}_2 \in \mathbb{R}^{d \times d}$ is the mapping matrix and $\mathbf{b} \in \mathbb{R}^{2d}$ is the weights for the user-item interaction attention layer. By employing the softmax to all the item neighbors, we derive the attention weights for item neighbors as

$$\beta_{nk} = \text{softmax}_k\left(v_{nk}\right) = \frac{\exp\left(v_{ni}\right)}{\sum_{i=1}^{K} \exp\left(v_{ni}\right)}, \tag{5}$$

where β_{nk} denotes the attention weights for items normalized over all neighboring items, next, we present the relational aggregation for both user neighbors and item neighbors.

4.2.2 Relational Graph Aggregation. Inferring the center user embeddings requires aggregating both neighbor user nodes and neighbor item nodes with their associated attention weights, which are formulated as follows:

$$\mathbf{h}_u^{(n)} = \sum_{p=1}^{P} \alpha_{np} \mathbf{W}_h \mathbf{e}_{u_p}, \quad \mathbf{h}_t^{(n)} = \sum_{k=1}^{K} \beta_{nk} \mathbf{W}_h \mathbf{e}_{t_k}, \tag{6}$$

where $\mathbf{W}_h \in \mathbb{R}^{d \times d}$ is the linear mapping weight matrix, and $\mathbf{h}_u^{(n)}, \mathbf{h}_t^{(n)} \in \mathbb{R}^d$ denote the hidden embeddings for aggregating user neighbors and item neighbors, respectively. The general aggregation process is illustrated in Figure 2(b). Intuitively, we should aggregate hidden embeddings and the center node embedding to infer the embedding of u_n. However, social relation, user-item interactions and center node embedding are not equally contributed to the learning process [61]. We should also handle the heterogeneity during the aggregation step. Therefore, we propose to use three relation vectors to preserve their semantics, i.e., $\mathbf{v}_u, \mathbf{v}_t, \mathbf{v}_s \in \mathbb{R}^d$ preserving the social semantics, user-item semantics and center node itself semantics respectively. To be more specific, we concatenate hidden embeddings with their relation vectors and employing the self-attention mechanism to learn the weights for aggregation

$$\gamma_u = \frac{\exp\left(\mathbf{c}^\top \left[\mathbf{h}_u^{(n)} \| \mathbf{v}_u\right]\right)}{\exp\left(\mathbf{c}^\top \left[\mathbf{h}_u^{(n)} \| \mathbf{v}_u\right]\right) + \exp\left(\mathbf{c}^\top \left[\mathbf{h}_t^{(n)} \| \mathbf{v}_t\right]\right) + \exp\left(\mathbf{c}^\top \left[\mathbf{h}_s^{(n)} \| \mathbf{v}_s\right]\right)}, \tag{7}$$

$$\gamma_t = \frac{\exp\left(\mathbf{c}^\top \left[\mathbf{h}_t^{(n)} \| \mathbf{v}_t\right]\right)}{\exp\left(\mathbf{c}^\top \left[\mathbf{h}_u^{(n)} \| \mathbf{v}_u\right]\right) + \exp\left(\mathbf{c}^\top \left[\mathbf{h}_t^{(n)} \| \mathbf{v}_t\right]\right) + \exp\left(\mathbf{c}^\top \left[\mathbf{h}_s^{(n)} \| \mathbf{v}_s\right]\right)}, \tag{8}$$

$$\gamma_s = \frac{\exp\left(\mathbf{c}^\top \left[\mathbf{h}_s^{(n)} \| \mathbf{v}_s\right]\right)}{\exp\left(\mathbf{c}^\top \left[\mathbf{h}_u^{(n)} \| \mathbf{v}_u\right]\right) + \exp\left(\mathbf{c}^\top \left[\mathbf{h}_t^{(n)} \| \mathbf{v}_t\right]\right) + \exp\left(\mathbf{c}^\top \left[\mathbf{h}_s^{(n)} \| \mathbf{v}_s\right]\right)}, \tag{9}$$

where γ_u, γ_t, and γ_s are the attention weights for hidden user neighbors embedding, hidden item neighbor embedding, and center node itself embedding, respectively. $\mathbf{c} \in \mathbb{R}^{2d}$ is the weight vector for the attention layer. Given that, we infer the node embeddings of u_n as

$$\mathbf{e}_{u_n}^* = \gamma_s \mathbf{e}_{u_n} + \gamma_u \mathbf{h}_u^{(n)} + \gamma_t \mathbf{h}_t^{(n)}, \tag{10}$$

where $\mathbf{e}_{u_n}^*$ is the local user embedding for prediction. The aggregation is illustrated in Figure 3. As such, local clients preserve their user embeddings, which tackles the personalization problem. Next, we will use the learned embeddings and downloaded item embeddings to make a prediction.

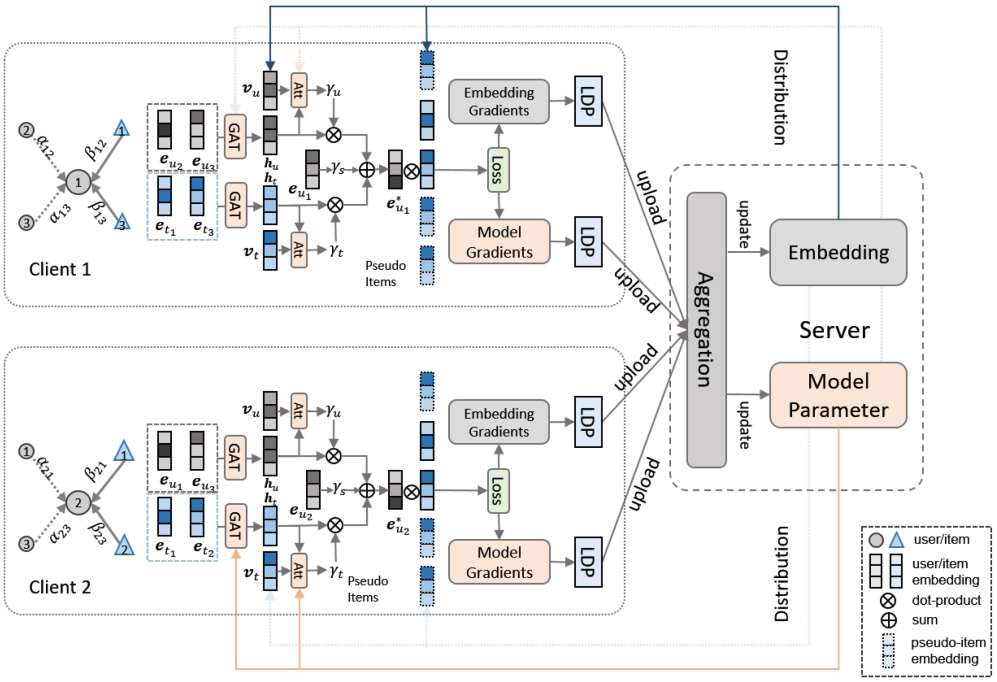


Fig. 3. The framework of FeSoG. For simplicity and without loss of generality, we present a two-client scenario. In each client, we use the local GAT layer to infer node embeddings and adopt the attention layer to aggregate social neighbors and item neighbors. Then, we sample a set of pseudo items bundled with local data to calculate the loss and gradients. Both the embedding gradients and model gradients are uploaded to the server for aggregation after LDP operation.

4.2.3 Prediction. To predict the local item ratings, we adopt the dot-product between the inferred user embedding and item embeddings. For a user u and an item t with embedding $\mathbf{e}_u^*$ and $\mathbf{e}_t$, respectively, the rating $\mathbf{R}_{ut}$ is

$$\hat{\mathbf{R}}_{ut} = \mathbf{e}_u^* \cdot \mathbf{e}_t, \tag{11}$$

where $\cdot$ denotes the dot-product operation. We use the local user-item rating values to optimize the prediction by employing the **Root Mean Squared Error** (**RMSE**) between the predicted score $\hat{\mathbf{R}}_{ut}$ and the ground-truth rating score $\mathbf{R}_{ut}$:

$$\mathcal{L}_u = \sqrt{\frac{\sum_{t \in \mathcal{T}^{(u)}} (\mathbf{R}_{ut} - \hat{\mathbf{R}}_{ut})^2}{|\mathcal{T}^{(u)}|}}, \tag{12}$$

where $\mathcal{T}^{(u)}$ denotes the rated items of user u, and $\mathcal{L}_u$ is the local loss for user u. Recall that each user is associated with a client. This loss function will be used to calculate the gradient for clients. Then, the gradients are collected from multiple clients to the server for further updates. However, directly uploading gradients leads the user-item interaction data to be vulnerable [4, 53]. It urges us to design a privacy protection mechanism regarding the gradients, which will be introduced next.

4.3 Privacy Protection

This section introduces two techniques to protect the local user-item interaction data when uploading the gradients: dynamic **Local Differential Privacy** (**LDP**) and pseudo-item labeling.

4.3.1 Local Differential Privacy. According to FedMF [4], the user's rating information can be inferred if given the gradients of a user uploaded in two continuous steps. Though our case is more complicated than in FedMF, it is still problematic if directly optimizing the local data and uploading the gradients. Moreover, for embedding gradients, only items with ratings in a local client have non-zero gradients to upload to the server. FedMF [4] proposes using encryption for the gradients so that the server cannot inverse the encoding process. However, it requires generating the public key and secret key and introducing additional computation for encryption. Additionally, to solve the zero-gradient for non-rated items, FedGNN [53] proposes to sample pseudo-interacted items and add Gaussian noise with the same mean and variance as the ground-truth items to their gradients. Another technique used broadly is the LDP module [7, 39, 41]. It adopts clipping the local gradients based on their L∞-norm with a threshold δ and applies a LDP module with zero-mean Laplacian noise to the unified gradients to achieve privacy protection.

To be more specific, we instantiate the item embedding gradients, user embedding gradients, and model gradients from client n as $\mathbf{g}_t^{(n)}$, $\mathbf{g}_u^{(n)}$, and $\mathbf{g}_m^{(n)}$, respectively. Combining them gives the gradients as $\mathbf{g}^{(n)} = \{\mathbf{g}_t^{(n)}, \mathbf{g}_u^{(n)}, \mathbf{g}_m^{(n)}\} = \frac{\partial \mathcal{L}_u}{\partial \Theta}$, where Θ denotes all trainable parameters. Then, the LDP is formulated as

$$\tilde{\mathbf{g}}^{(n)} = \text{clip}(\mathbf{g}^{(n)}, \delta) + \text{Laplacian}(0, \lambda), \tag{13}$$

where $\tilde{\mathbf{g}}^{(n)}$ is the randomized gradients, $\text{clip}(x, \delta)$ denotes limiting x with the threshold δ, and $\text{Laplacian}(0, \lambda)$ is the Laplacian noise with 0 mean and λ strength. However, a constant noise strength is inappropriate when dealing with gradients at different magnitudes. The gradient magnitude of different parameters varies during training. Hence, we propose to add dynamic noise based on the gradient, which is formulated as follows:

$$\tilde{\mathbf{g}}^{(n)} = \text{clip}(\mathbf{g}^{(n)}, \delta) + \text{Laplacian}(0, \lambda \cdot \text{mean}(\mathbf{g}^{(n)})), \tag{14}$$

4.3.2 Pseudo-Item Labeling. Based on existing work, we propose a new privacy protection module, which is advantageous as it can protect the training gradients and enhance the model with more robustness. In a local client, before calculating the training loss, we first sample q items not in the neighbor items, which are the pseudo-items in Figure 3, denoting as $\tilde{\mathcal{T}}^{(u)} = \{\tilde{t}_1^{(u)}, \tilde{t}_2^{(u)}, \ldots, \tilde{t}_q^{(u)}\}$. Then, we use the local model to predict the ratings for these pseudo items. The predicted ratings are rounded to be the pseudo ratings. Hence, the loss in Equation (12) is changed to

$$\tilde{\mathcal{L}}_u = \sqrt{\frac{\sum_{t \in \mathcal{T}^{(u)} \cup \tilde{\mathcal{T}}^{(u)}} (\mathbf{R}_{ut} - \hat{\mathbf{R}}_{ut})^2}{|\mathcal{T}^{(u)}|}}. \tag{15}$$

Note that compared with Equations (12) and (15) is calculated from both the true interacted items and pseudo items. The ground-truth ratings for pseudo items are the rounded predicted score. The difference between the predicted score and the rounded one contributes to the gradients for those pseudo items. Here, we assume that $\mathbf{R}_{ut} \in \mathbb{N}$, while $\hat{\mathbf{R}}_{ut} \in \mathbb{R}$. The gradients derived from Equation (15) contain both ground-truth rating information and the pseudo item rating information, which prevents the data leakage problem. Additionally, the pseudo labels of items provide additional rating information, which can alleviate the cold-start issue of the data. Intuitively, this technique works as a data augmentation method [26, 28, 59]. We sample those pseudo items and view the difference between rounded ratings with a predicted rating as the randomness, which enhances the robustness of the local model.

4.4 Optimization

In this section, we first present the optimization process of the FeSoG framework before we present the pseudo-code of the algorithm.

4.4.1 Gradient Collection for Optimization. The server in FeSoG collects the gradients uploaded from clients to update both the model parameters and embeddings, which collaboratively optimize the model. Recall that the gradient from client n is $\tilde{\mathbf{g}}^{(n)}$ and the parameter is Θ, which includes Θ_m, Θ_t, and Θ_u indicating model parameters, item embedding, and user embedding, separately. In each round, the server builds a connection with a batch (e.g., 128) of clients, denoted as $\mathcal{N}$. It first sends the current model parameters Θ_m and embeddings Θ_e to those clients. Then, it aggregates local gradients from those clients as follows:

$$\bar{\mathbf{g}}_m = \frac{\sum_{n\in\mathcal{N}} |\mathcal{R}_n| \cdot \tilde{\mathbf{g}}_m^{(n)}}{\sum_{n\in\mathcal{N}} |\mathcal{R}_n|}, \quad \bar{\mathbf{g}}_t = \frac{\sum_{n\in\mathcal{N}} |\mathcal{R}_n| \cdot \tilde{\mathbf{g}}_t^{(n)}}{\sum_{n\in\mathcal{N}} |\mathcal{R}_n^t|}, \quad \bar{\mathbf{g}}_u = \frac{\sum_{n\in\mathcal{N}} |\mathcal{R}_n| \cdot \tilde{\mathbf{g}}_u^{(n)}}{\sum_{n\in\mathcal{N}} |\mathcal{R}_n^u|}, \tag{16}$$

where $\mathcal{R}_n$ is the total number interaction for calculating the gradients, including both the real interactions and pseudo interactions. $\mathcal{R}_n^t$ and $\mathcal{R}_n^u$ indicate interactions involving item t and user u, separately. Intuitively, $\bar{\mathbf{g}}_m$, $\bar{\mathbf{g}}_t$, $\bar{\mathbf{g}}_u$ are weighted average of the gradients from clients. After aggregation, the server updates the parameter Θ with gradient descent as

$$\Theta^* = \Theta - \eta \cdot \bar{\mathbf{g}}, \tag{17}$$

where η is the learning rate. This learning process is operated multiple rounds until convergence.

4.4.2 Algorithm. The pseudo-code of the algorithm of FeSoG is presented in Algorithm 1. The inputs are consist of the training hyper-parameters such as the embedding size d and learning η. Additionally, the client data should also be given, i.e., the clients local graphs $\{\mathcal{G}_n|_{n=1}^N\}$. Though the target is to predict item ratings, we output the parameters Θ and the local inferred embeddings $\{\mathbf{e}_{u_n}^*|_{n=1}^N\}$, which is sufficient for clients to predict ratings. In the algorithm, the line 2 to the line 6

ALGORITHM 1: FeSoG (Federated Social Recommendation with Graph Neural Network)

Input : Embedding Size, learning rate: d, η
Total number of clients, items: N, M, T
The number of pseudo items: p
LDP parameter: δ, λ
Clients local graph: $\{\mathcal{G}_n|_{n=1}^N\}$

Output : Model parameters and embeddings Θ; Local client embeddings $\{\mathbf{e}_{u_n}^*|_{n=1}^N\}$

1 Initializing Θ;
2 **while** *not converge* **do**
3 sampling a fractions of clients $\mathcal{N}$; **for** $n \in \mathcal{N}$ **do**
4 $\mathbf{g}^{(n)}, |\mathbf{R}_n|$ = `ClientUpdate`(n, Θ); `// collecting gradients from clients`
5 $\bar{\mathbf{g}}^{(n)} \leftarrow$ Equation (16); `// averaging gradients from clients`
6 $\Theta = \Theta - \eta \cdot \bar{\mathbf{g}}^{(n)}$; `// updating parameters`
7 **Function** `ClientUpdate`(n, Θ)**:**
8 downloading Θ from server;
9 $\mathbf{e}_{u_n}^* \leftarrow$ Equation (10); `// local user embedding inference`
10 sampling p pseudo items;
11 calculating the ratings of those pseudo items using Equation (11); `// pseudo-labelling`
12 $\tilde{\mathcal{L}}_n \leftarrow$ Equation (15);
13 $\mathbf{g}^{(n)} = \frac{\partial \tilde{\mathcal{L}}_u}{\partial \Theta}$; `// computing the gradients`
14 $\tilde{\mathbf{g}}^{(n)} \leftarrow$ Equation (13); `// LDP for gradients`
15 **return** $\tilde{\mathbf{g}}^{(n)}, p + |\mathcal{T}^{(u)}|$; `// return gradients and the number of interactions`

Table 3. Statistics of Datasets

Dataset	Ciao	Epinions	Filmtrust
Users	7,317	18,069	874
Items	104,975	261,246	1,957
# of ratings	283,320	762,938	18,662
Rating density	0.0369%	0.0162%	1.0911%
# of social connections	111,781	355,530	1,853
Social connection density	0.2088%	0.1089%	0.2426%

is the loop operated on the server, which sends parameters to clients and collects their gradients for updating. The function ClientUpdate() is the operation on local devices. It downloads the parameters to infer the local user embeddings (line 8). Then, the pseudo items are sampled (line 10). Pseudo-labeling and LDP are combined (lines 10–14) to protect the gradients from privacy leakage. This function returns the gradients and the number of interactions (line 15) for the server to collect.

5 EXPERIMENTS

In this section, we conduct experiments to evaluate the effectiveness of FeSoG. We will answer the following **Research Questions** (**RQs**):

— **RQ1**: Does FeSoG outperform existing methods in social recommendation?
— **RQ2**: What is the impact of the hyper-parameters in FeSoG?
— **RQ3**: Are those components in FeSoG necessary?

5.1 Experimental Setup

5.1.1 Datasets. In this article, we adopt three commonly used social recommendation datasets to conduct experimental analyses, which are Ciao, Epinions [45–47], and Filmtrust [13]. Ciao and Epinions[2] [45–47] are crawled from shopping website. Both datasets contain user rating scores on items and trust links between users as social relations. Each user can give an integer score in $\{1, 2, 3, 4, 5\}$ to rate an item, where 1 indicates least like, while 5 represents most. Filmtrust[3] [13] is built from online film rating website and the trust relationship between users. The rating scale ranges from 1 to 8. Social relations are also the trust links between users in these datasets. Data statistics are shown in Table 3. In our FSRS scenario, each user is treated as a local client, and the user's interactions are local privacy data on the device. The global graph information is transferred from user embeddings.

5.1.2 Baselines. We adopt three types of baselines for comparison: traditional MF -based methods for social recommendation, recent GNN-based methods for social recommendation, and federated learning frameworks. MF-based and GNN-based methods are based on centralized learning, which is unable to protect user privacy. Federated learning methods are not able to handle the fusion of local social information and rating information.

MF-based methods

— SoRec [32]: It co-factorizes user-item rating matrix and user-user social matrix.
— SoReg [33]: It develops a social regularization with social links to regularize on MF.
— SocialMF [18]: Compared with SoReg, social MF also considers social trust propagation.

[2] https://www.cse.msu.edu/~tangjili/datasetcode/truststudy.html.
[3] https://guoguibing.github.io/librec/datasets.html.

— CUNE [64]: Collaborative user network embedding assumes users hold implicit social links from each other, and it tries to extract semantic and reliable social information by graph embedding method.

GNN-based methods

— GCMC+SN [2]: GCMC is a GNN-based method. User nodes are initialized as vectors learned by node2vec [12] from the social graph to obtain social information. The dense representation learned upon the social graph can include more information than the random initialized feature.
— GraphRec [9]: Graph recommendation uses GNN to learn user embedding and item embedding from their neighbors and uses several fully connected layers as the rating predictor.
— ConsisRec [61]: It is the **state-of-the-art** (**SotA**) method in social recommendation. ConsisRec modifies GNN to mitigate the inconsistency problems in social recommendation.

Federated learning methods

— FedMF [4]: It separates the MF computation to different users and uses an encryption method to avoid information leakage.
— FedGNN [53]: Federated GNN is the SotA federated recommendation method. It adopts local differential privacy methods to protect user's interaction with items.

5.1.3 Evaluation Metrics. To evaluate the performance and compare, we adopt **Mean Absolute Error** (**MAE**) and RMSE to measure the model performance because they are the most commonly used metrics in social recommendation. Smaller values of both two metrics indicate better performance in the test data. The two metrics are calculated as follows:

$$\text{MAE} = \frac{\sum_{n=1}^{N} \sum_{t \in \mathcal{T}^{(n)}} |\mathbf{R}_{nt} - \hat{\mathbf{R}}_{nt}|}{\sum_{n=1}^{N} |\mathcal{T}^{(n)}|}, \tag{18}$$

$$\text{RMSE} = \sqrt{\frac{\sum_{n=1}^{N} \sum_{t \in \mathcal{T}^{(n)}} (\mathbf{R}_{nt} - \hat{\mathbf{R}}_{nt})^2}{\sum_{n=1}^{N} |\mathcal{T}^{(n)}|}}, \tag{19}$$

where $\mathbf{R}_{nt}$ and $\hat{\mathbf{R}}_{nt}$ are the true rating value and predicted rating value of user n for item t, respectively. N is the total number of users for testing. $\mathcal{T}^{(n)}$ denotes the rated items for user n. Again, the evaluation is conducted on devices locally since the server has no access to the local privacy data. The lower MAE and RMSR both indicate better performance.

5.1.4 Experimental Setups. The ratings in each dataset are randomly split into training set (60%), validation set (20%), and test set (20%). Hyper-parameters are tuned based on the validation performance. Then, we report the final performance on the test dataset. In all experiments, we initialize the parameters with standard Gaussian distribution. For the LDP technique used in FeSoG, the gradient clipping threshold is set to 0.3, and the strength of Laplacian noise is set to 0.1. Other hyper-parameters are tuned based on grid searching. The number of pseudo interacted items p is searched in $\{10, 50, 100, 500, 1000\}$. Embedding size d is tuned from $\{4, 8, 16, 32, 64\}$. User batch size in each training round is searched in $\{16, 32, 64, 128, 256\}$. Learning rate η is searched in $\{0.1, 0.05, 0.01\}$. Training is stopped if RMSE on the validation set does not improve for five successive validations.

5.2 Overall Comparison (RQ1)

In this section, we conduct the overall comparison of different models. The experimental results are shown in Table 4, which are categorized into three groups. We have the following observations:

Table 4. Experiment Results Compared with Baseline Methods

Method	Ciao		Epinions		Filmtrust	
	RMSE	MAE	RMSE	MAE	RMSE	MAE
SoRec	1.2024	0.8693	1.3389	1.0618	1.8094	1.4529
SoReg	1.0066	0.7595	1.0751	0.8309	1.7950	1.4413
SocialMF	1.0013	0.7535	1.0706	0.8264	1.8077	1.4557
GCMC+SN	1.0301	0.7970	1.1070	0.8480	1.8025	1.4325
GraphRec	1.0040	0.7591	1.0799	0.8219	1.6775	1.3194
CUNE	1.0002	0.7591	1.0681	0.8284	1.7675	1.4178
ConsisRec	0.9722	0.7394	1.0495	0.8046	1.7148	1.3093
FedMF	2.4216	2.0792	2.0685	1.5254	2.795	2.1713
FedGNN	2.02	1.58	1.8346	1.4238	2.13	1.65
FeSoG	**1.9136**	**1.4937**	**1.7969**	**1.3847**	**2.0942**	**1.5855**
Improvement	5.26%	5.46%	2.05%	2.74%	1.68%	3.9%

The best federated learning results are in bold, and the best results for non-federated learning methods are underlined. Improvement indicates the percent that FeSoG improves against the second-best federated learning result.

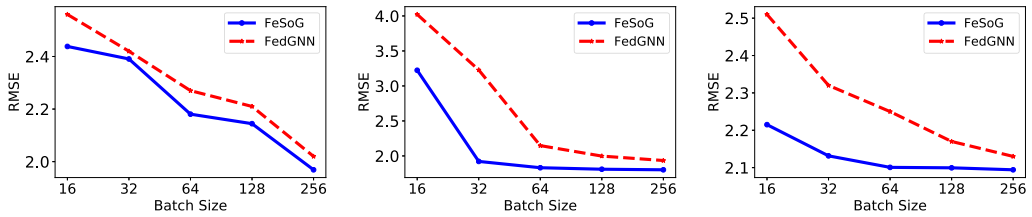
— FeSoG significantly outperforms the SOTA federated recommender systems in all datasets. Compared with FedGNN, FeSoG achieves on average 2.99% and 4.03% relative improvements on RMSE and MAE, respectively. Several advantages of FeSoG support its superiority: (1) social information helps the recommendation, which the improvement can demonstrate over FedMF; (2) the relational graph attention and aggregation can effectively integrate both user-item interactions and social information; and (3) the local pseudo-item sampling technique enhances the performance.

— The GNN-based models perform better than those MF-based models. ConsisRec is the SOTA-GNN model that employs relation attention and consistent neighbor aggregation, which leads to its best performance. Compared with SocialMF, which is the best MF-based method, ConsisRec achieves in average 3.47% and 5.27% relative improvements in RMSE and MAE, respectively. GNN-based models are better as they can directly model structure information and simultaneously aggregate social and user-item interactions. Between the two federated learning baselines, FedGNN also significantly outperforms FedMF, which again supports the claims that GNN-based models are better than MF-based methods. FeSoG is also based on GNN aggregation. Its local GNN aggregation employs relation attention, which leads to its better performance against FedGNN.

— Federated learning impairs the performance compared with centralized learning. Even a simple GCMC+SN model is better than the FedGNN model. There are two reasons: On one hand, to achieve privacy protection, the federated learning framework has no access to the local data, limiting its capacity to model the global structures. According to [9, 27], the core of graph embedding is to aggregate high-order neighbors and select informative contexts. On the other hand, the local gradients are protected by adding random noise. Though it theoretically will not hurt the performance in expectation, it still prevents the server from receiving qualitative gradients from clients. We should find a tradeoff between performance and privacy protection. This observation also brings opportunities for federated learning research.

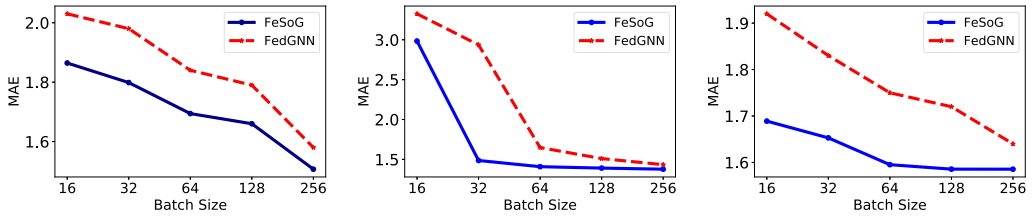
5.3 Sensitivity Analysis (RQ2)

In this section, we emphasize on analyzing the impacts of those hyper-parameters involving in FeSoG and some other baselines. We include user batch size $|\mathcal{N}|$ as in the line 3 of Algorithm 1,



(a) Ciao (b) Epinions (c) Filmtrust

Fig. 4. RMSE performance with respect to user batch size on three datasets.



(a) Ciao (b) Epinions (c) Filmtrust

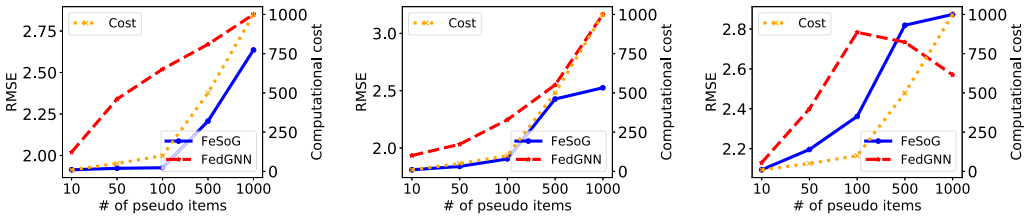
Fig. 5. MAE performance with respect to user batch size on three datasets.

embedding size d, learning rate η, the number of pseudo items p, local differential privacy parameter δ, and λ.

5.3.1 User Batch Size. In this section, we analyze the impact of user batch size. Intuitively, choosing a small user batch size increases the communication rounds for the server to train a model. However, it is unclear how it affects the prediction performance. We report the performance of FedGNN and FeSoG with respect to RMSE and MAE across three datasets, which is illustrated in Figures 4 and 5, respectively. We have the following observations:

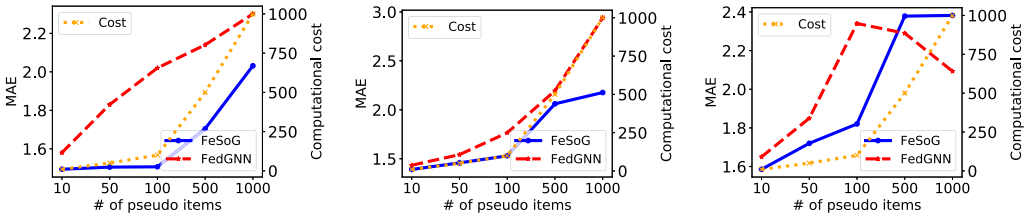
— FeSoG performs better than FedGNN. On all three datasets, the RMSE of FeSoG is consistently lower than FedGNN, which results from its powerful embedding ability of relational local GNN.
— The performance of FeSoG becomes better with the increase of user batch size across datasets. With a larger user batch size, the server can obtain a more accurate global information estimation, which leads to a better performance. However, in practice, aggregating more users at each training step would lead to more computational cost and more time to converge.

5.3.2 Number of Pseudo Items. Pseudo items are sampled to protect the gradients from privacy leakage. Sampling more pseudo items requires more computational cost as more ratings should be predicted. However, it is unclear how many samples should be selected to achieve satisfying results. Hence, we conduct experiments on three datasets to study the impacts of the number of sampled pseudo items. We also compare FedGNN, which samples a set of negative items and assigns them with random gradients. The influence of the number of pseudo items on three datasets with respect to RMSE and MAE is reported in Figures 6 and 7. Besides the performance value, we also present the computational cost with respect to the number of sampled pseudo items. We have the following observations:



(a) Ciao (b) Epinions (c) Filmtrust

Fig. 6. RMSE performance with respect to different pseudo item numbers on three datasets.

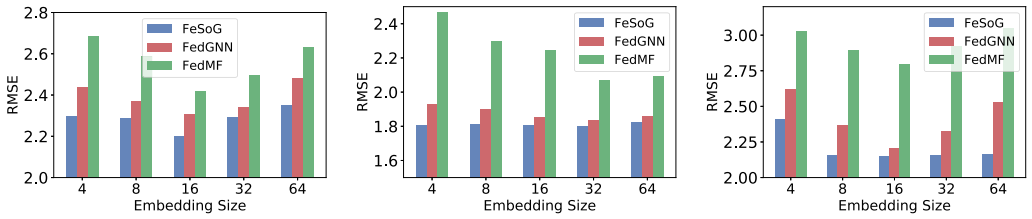


(a) Ciao (b) Epinions (c) Filmtrust

Fig. 7. MAE performance with respect to different pseudo item numbers on three datasets.

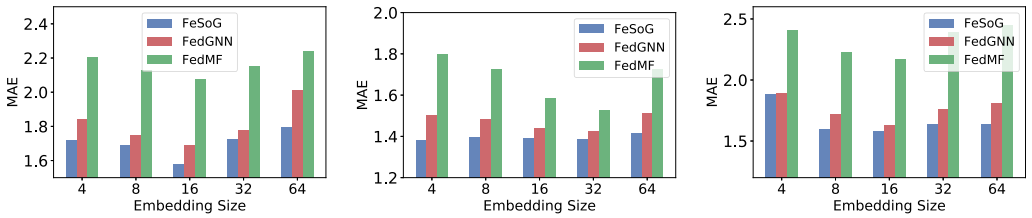
— FeSoG yields better performance compared with FedGNN. On Ciao and Epinions datasets, the performance of FeSoG and FedGNN gets worse with the increase of pseudo items. However, the value of FeSoG is much lower than FedGNN. It suggests that our pseudo item sampling and the pseudo labeling techniques are robust. Therefore, we can conclude that the pseudo item sampling in FeSoG can protect privacy data and enhance the training process.
— If increasing the number of pseudo items, the error value all increases for both FedGNN and FeSoG on Ciao and Epinions datasets, which results from more noise caused by pseudo items. However, on the Filmtrust dataset, the error value of FedGNN first increases and then drops. This is because FedGNN generates gradients of pseudo items from the same Gaussian distribution, and at the same time Filmtrust dataset only has 1,957 items. So when sampling more than 500 pseudo items, the gradient of pseudo items from different users would counteract with each other to reduce noise impact. FeSoG generates gradients of pseudo items by user-specific labeling, which does not have this characteristic.
— We should find a tradeoff between pseudo-item sampling and model performance. As illustrated in Figures 6 and 7, if increasing the number of pseudo items, the extra computational cost increases linearly and the server would be harder to infer the true interacted items. At the same time, prediction accuracy would become worse. So we should find a suitable pseudo item number to balance privacy protection and model performance. For example, 100 pseudo items for Ciao dataset would be an appropriate setting because the model performance does not deteriorate much when the number of pseudo items is lower than 100.

5.3.3 Embedding Size. This section studies the performance with respect to the embedding size. The values of RMSE and MAE on three datasets are reported in Figures 8 and 9, respectively. For comparison, we also select two representative baselines, which are FedGNN and FedMF. We have the following observations:



(a) Ciao (b) Epinions (c) Filmtrust

Fig. 8. RMSE performance with respect to different embedding sizes d on three datasets.



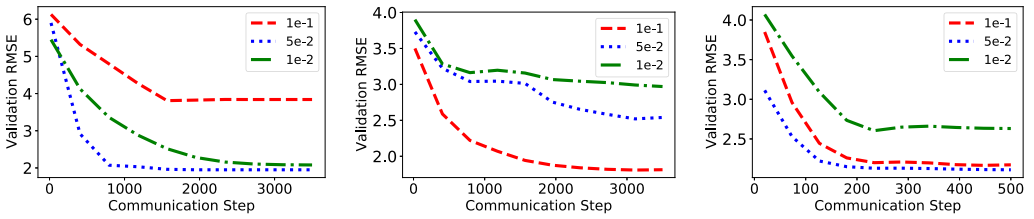
(a) Ciao (b) Epinions (c) Filmtrust

Fig. 9. MAE performance with respect to different embedding sizes d on three datasets.

— The proposed FeSoG consistently outperforms other federated recommender system methods across all embedding sizes. This observation suggests that FeSoG can learn informative structures from local graphs. Compared with FedMF, FedGNN demonstrates a much better performance, which justifies the necessity to employ GNN for graph embedding.
— FeSoG has lower fluctuations as embedding sizes change compared with the other two baselines. It indicates that FeSoG is more robust than FedGNN on different embedding sizes, demonstrating the effectiveness of using pseudo-item protection to enhance the training.
— Furthermore, suitable embedding sizes are crucial for different datasets to achieve satisfying performance. All federated learning methods follow the same trend and obtain the best performance in either $d = 16$ or $d = 32$. With a smaller embedding size (e.g., $d = 4$), the model has insufficient representation ability. However, with large embedding sizes (e.g., $d = 64$), it may cause the overfitting issue due to limited data.

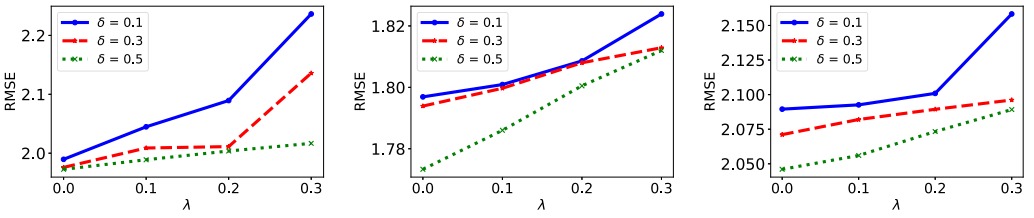
5.3.4 Learning Rate. Learning rate affects the number of communication rounds for the convergence of a federated learning framework. Intuitively, fewer communication steps can decrease the number of communication times between the server and clients, which implies less risks of information leakage. We investigate the impact of learning rate with respect to the training loss of FeSoG and report the results in Figure 10. The learning rather is selected from $\{0.1, 0.05, 0.01\}$. We have the following observations:

— The training of FeSoG is smooth. The loss on three datasets all converges smoothly when increasing the number of communications steps. It suggests that the federated learning framework can effectively transfer informative gradients for the FeSoG to converge.
— Different datasets prefer different learning rates. For example, on the Epinions dataset, the best learning rate $\eta = 0.1$. While both Ciao and Filmtrust datasets converge the fastest when the learning rate $\eta = 0.05$. Learning rate has a critical influence on different datasets.



(a) Ciao (b) Epinions (c) Filmtrust

Fig. 10. Loss curves for different learning rate η on three datasets.



(a) Ciao (b) Epinions (c) Filmtrust

Fig. 11. RMSE performance with respect to different δ and λ on three datasets.

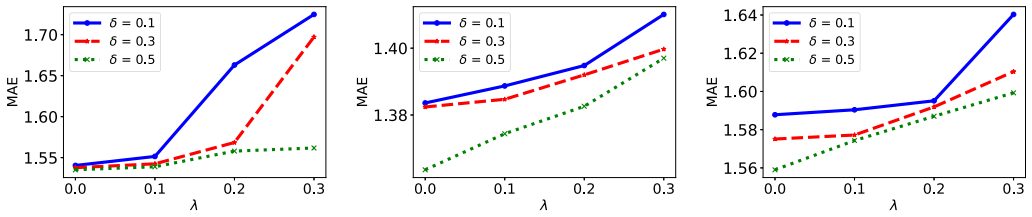
5.3.5 Local Differential Privacy Parameters δ and λ. This section studies the correlations between model performance and the LDP module. It contains two parameters: gradient clip threshold δ and Laplace noise strength λ, as shown in Equation (14). Intuitively, this module alters gradients by injecting noise to gradients to protect the user's privacy. The injected gradients contribute to the training because the server also aggregates them for updating. Therefore, we conduct experiments to study the model performance with respect to the variations of these two parameters, which are shown in Figures 11 and 12 for RMSE and MAE, respectively. We have the following observations:

— With a fixed λ, FeSoG performs better when increasing δ. The reason is that a large δ tends to clip less gradients. Therefore, the aggregated gradient information would be more accurate to reflect the true gradients.
— With fixed δ, FeSoG performs worse when increasing λ. It is because a larger λ injects more substantial Laplace noise to the model gradient. Hence, the gradient learned from data would be overwhelmed by generated noises. Thus, a smaller λ is preferred.
— There is a tradeoff in selecting optimal values. Although larger δ or smaller λ lead to better performance, it would increase the risk of privacy leakage. If δ is infinitely large and λ is 0, the server can revert the interactions by checking uploaded gradients from clients. Therefore, we should choose an optimal pair to achieve acceptable performance with a small enough privacy budget.

5.4 Ablation Study (RQ3)

In this section, we conduct an ablation study to analyze those components in FeSoG to validate their effectiveness. We create three other variants of FeSoG:

— **sharing GAT layer:** FeSoG applies different GAT layers to learn the attention weights for social neighbors and item neighbors. This variant shares the GAT layer for all neighbors.



(a) Ciao (b) Epinions (c) Filmtrust

Fig. 12. MAE performance with respect to different δ and λ on three datasets.

Table 5. Ablation Study on FeSoG

Variant	Ciao		Epinions		Filmtrust	
	RMSE	MAE	RMSE	MAE	RMSE	MAE
sharing GAT layer	2.0262	1.5863	1.8046	1.3876	2.148	1.5939
relative difference	5.8%	6.2%	0.43%	0.21%	2.57%	0.53%
w/o relational vector	2.1545	1.7023	1.8044	1.3903	2.1379	1.6441
relative difference	12.58%	13.96%	0.42%	0.40%	2.09%	3.7%
w/o pseudo items	**1.9026**	**1.4849**	**1.7053**	**1.3021**	**2.0825**	**1.5798**
relative difference	−0.57%	−0.58%	−5.1%	−5.97%	−0.56%	−0.36%
FeSoG	<u>1.9136</u>	<u>1.4937</u>	<u>1.7969</u>	<u>1.3847</u>	<u>2.0942</u>	<u>1.5855</u>

The relative difference represents the performance difference between the corresponding variant and FeSoG. A positive difference indicates worse performance, while negative ones indicate better performance.
Bold and <u>underline</u> values denote the best and second-best.

Hence, we only have one GAT layer to learn weight. Note that this variant also employs the relational vector during aggregation.

— **w/o relational vectors:** this variant ignores the social relation and user-item relation vectors during aggregation. As such, it directly aggregates all the neighbors with their associated attention weights. Since the attention weights are learned separately for user neighbors and item neighbors, we should normalize those weights for all neighbors.

— **w/o pseudo items:** this variant is a special case of Section 5.3.2 where the number of the pseudo item is 0. It cannot protect the user privacy data from a protection perspective because. Without pseudo items, we only use the true interacted items that have non-zero gradients. However, we also include this variant for a comprehensive study.

The performance comparison of these methods on three datasets is reported in Table 5. We also present the corresponding difference between FeSoG and the variants in a group. We have the following observations:

— We should employ different GAT layers for users neighbors and item neighbors. Compared with FeSoG, sharing GAT layer has worse performance. On the Filmtrust dataset, it has 2.57% and 0.53% relative difference on RMSE and MAE, respectively. On the Epinions dataset, it has 0.43% and 0.21% relative difference on RMSE and MAE, respectively. The worst performance of this variant is on Ciao, which has 5.8% and 6.2% relative difference on RMSE and MAE, respectively.

— We should apply the relational vectors for aggregation. Compared with FeSoG, without relational vectors has worse performance. On the Epinions dataset, it has 0.42% and 0.40% relative difference on RMSE and MAE, respectively. On the Filmtrust dataset, it has 2.09%

and 3.7% relative difference on RMSE and MAE, respectively. The worst performance of this variant is on Ciao, which has 12.58% and 13.96% relative difference on RMSE and MAE, respectively.

— Sampling pseudo items always worsen performance. However, compared with FeSoG, without pseudo items is unable to protect the privacy because the server can easily infer the true interacted items by checking which item gradient is not 0.

6 CONCLUSION AND FUTURE WORK

In this article, we propose a new federated learning framework, FeSoG, for social recommendation. It decentralizes the data storage compared with existing social recommender systems. Moreover, it comprehensively fuses the local user privacy data in clients and uses a server to train an FSRS collaboratively. We address three challenges in designing this model: the heterogeneity of the data, the personalization requirements of the local modeling, and privacy protection for communication. The components in FeSoG jointly tackle these challenges: The relational attention and aggregation of the local GNN distinguish social and item neighbors; The local user embedding inference preserves the personalizing information for clients; The pseudo-item labeling, as well as the dynamic LDP technique, protect the gradients from privacy data leakage. To verify the effectiveness of FeSoG, we conduct extensive experiments. The overall comparing experiments demonstrate that FeSoG significantly outperforms SOTA federated learning framework in solving social recommendation problems. Detailed sensitivity analysis regarding the hyper-parameters further justifies the efficacy of FeSoG infusing the social information and user-item interaction and preserving the user privacy data locally. Moreover, the ablation study by dropping the components in FeSoG demonstrates the necessity of our designing.

Though being effective in solving the social recommendation problem, there are still several future directions. Firstly, we randomly sample pseudo items and predict their pseudo labels to protect gradients. However, as the items may have their relations, we may investigate employing an adaptive sampling of items rather than randomly. For example, we may train a local reinforcement learning model to explore the non-interacted item, which can decrease the noise. Secondly, we train the relational GNN only by leveraging local data. It is also possible to extend the local graph to be a high-order graph. However, this requires data transferring among clients. To protect privacy, we may design a peer-to-peer communication of clients preserving the decentralized storage characteristics of a federated recommender system. Finally, we can study the efficiency of communication. Since it requires numerous communication rounds to train a federated learning framework, it will be satisfying to decrease the time for communication or increase the bandwidth for communication with large-scale clients.

REFERENCES

[1] Muhammad Ammad-Ud-Din, Elena Ivannikova, Suleiman A. Khan, Were Oyomno, Qiang Fu, Kuan Eeik Tan, and Adrian Flanagan. 2019. Federated collaborative filtering for privacy-preserving personalized recommendation system. arXiv:1901.09888. Retrieved from https://arxiv.org/abs/1901.09888.

[2] Rianne van den Berg, Thomas N. Kipf, and Max Welling. 2017. Graph convolutional matrix completion. arXiv:1706.02263. Retrieved from https://arxiv.org/abs/1706.02263.

[3] Yuwei Cao, Hao Peng, Jia Wu, Yingtong Dou, Jianxin Li, and Philip S. Yu. 2021. Knowledge-preserving incremental social event detection via heterogeneous GNNs. In *Proceedings of the Web Conference 2021*. 3383–3395.

[4] Di Chai, Leye Wang, Kai Chen, and Qiang Yang. 2020. Secure federated matrix factorization. *IEEE Intelligent Systems* 36, 5 (2020), 11–20.

[5] Chen Chen, Jingfeng Zhang, Anthony K. H. Tung, Mohan Kankanhalli, and Gang Chen. 2020. Robust federated recommendation system. arXiv:2006.08259. Retrieved from https://arxiv.org/abs/2006.08259.

[6] Yingtong Dou, Zhiwei Liu, Li Sun, Yutong Deng, Hao Peng, and Philip S. Yu. 2020. Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*. 315–324.

[7] Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. 2014. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. 1054–1067.

[8] Alireza Fallah, Aryan Mokhtari, and Asuman Ozdaglar. 2020. Personalized federated learning: A meta-learning approach. In *NeurIPS*.

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

[10] Wenqi Fan, Yao Ma, Qing Li, Jianping Wang, Guoyong Cai, Jiliang Tang, and Dawei Yin. 2019. A graph neural network framework for social recommendations. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*. ACM, 417–426.

[11] Adrian Flanagan, Were Oyomno, Alexander Grigorievskiy, Kuan Eeik Tan, Suleiman A. Khan, and Muhammad Ammad-Ud-Din. 2020. Federated multi-view matrix factorization for personalized recommendations. 12458 (2020), 324–347.

[12] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 855–864.

[13] G. Guo, J. Zhang, and N. Yorke-Smith. 2013. A novel Bayesian similarity measure for recommender systems. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*. 2619–2625.

[14] 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.

[15] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems*. 1024–1034.

[16] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. 2020. Heterogeneous graph transformer. In *Proceedings of the Web Conference 2020*. 2704–2710.

[17] Mohsen Jamali and Martin Ester. 2009. Trustwalker: A random walk model for combining trust-based and item-based recommendation. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 397–406.

[18] Mohsen Jamali and Martin Ester. 2010. A matrix factorization technique with trust propagation for recommendation in social networks. In *Proceedings of the 4th ACM Conference on Recommender Systems*. 135–142.

[19] Santosh Kabbur, Xia Ning, and George Karypis. 2013. Fism: Factored item similarity models for top-n recommender systems. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 659–667.

[20] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista A. Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaïd Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal, Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Hang Qi, Daniel Ramage, Ramesh Raskar, Mariana Raykova, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. 2019. Advances and open problems in federated learning. *Found. Trends Mach. Learn.* 14, 1–2 (2021), 1–210.

[21] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *Proceedings of the 5th International Conference on Learning Representations Conference Track Proceedings*. OpenReview.net.

[22] Jakub Konečnỳ, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. 2016. Federated learning: Strategies for improving communication efficiency. CoRR abs/1610.05492 (2016).

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

[24] Munan Li, Kenji Tei, and Yoshiaki Fukazawa. 2020. An efficient adaptive attention neural network for social recommendation. *IEEE Access* 8 (2020), 63595–63606.

[25] Peng Liu, Lemei Zhang, and Jon Atle Gulla. 2019. Real-time social recommendation based on graph embedding and temporal context. *International Journal of Human-Computer Studies* 121 (2019), 58–72.

[26] Ye Liu, Yao Wan, Lifang He, Hao Peng, and Philip S. Yu. 2020. KG-BART: Knowledge graph-augmented BART for generative commonsense reasoning. In *Proceedings of the AAAI Conference on Articial Intelligence*, Vol. 35. 6418–6425.

[27] Zhiwei Liu, Yingtong Dou, Philip S. Yu, Yutong Deng, and Hao Peng. 2020. Alleviating the inconsistency problem of applying graph neural network to fraud detection. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 1569–1572.

[28] Zhiwei Liu, Ziwei Fan, Yu Wang, and Philip S. Yu. 2021. Augmenting sequential recommendation with pseudo-prioritems via reversely pre-training transformer. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[29] Zhiwei Liu, Lin Meng, Jiawei Zhang, and Philip S. Yu. 2020. Deoscillated graph collaborative filtering. CoRR abs/2011.02100 (2020).

[30] Zhiwei Liu, Mengting Wan, Stephen Guo, Kannan Achan, and Philip S. Yu. 2020. Basconv: Aggregating heterogeneous interactions for basket recommendation with graph convolutional neural network. In *Proceedings of the 2020 SIAM International Conference on Data Mining*. SIAM, 64–72.

[31] 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.

[32] 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.

[33] Hao Ma, Dengyong Zhou, Chao Liu, Michael R. Lyu, and Irwin King. 2011. Recommender systems with social regularization. In *Proceedings of the 4th ACM International Conference on Web Search and Data Mining*. 287–296.

[34] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the Artificial Intelligence and Statistics*. PMLR, 1273–1282.

[35] Frank McSherry and Ilya Mironov. 2009. Differentially private recommender systems: Building privacy into the netflix prize contenders. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 627–636.

[36] Nan Mu, Daren Zha, Yuanye He, and Zhihao Tang. 2019. Graph attention networks for neural social recommendation. In *Proceedings of the 31st IEEE International Conference on Tools with Artificial Intelligence*. IEEE, 1320–1327.

[37] Hao Peng, Jianxin Li, Yangqiu Song, Renyu Yang, Rajiv Ranjan, Philip S. Yu, and Lifang He. 2021. Streaming social event detection and evolution discovery in heterogeneous information networks. *ACM Transactions on Knowledge Discovery from Data* 15, 5 (2021), 1–33.

[38] Hao Peng, Renyu Yang, Zheng Wang, Jianxin Li, Lifang He, Philip Yu, Albert Zomaya, and Raj Ranjan. 2021. Lime: Low-cost incremental learning for dynamic heterogeneous information networks. *IEEE Transactions on Computers*.

[39] Tao Qi, Fangzhao Wu, Chuhan Wu, Yongfeng Huang, and Xing Xie. 2020. Privacy-preserving news recommendation model learning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*. 1423–1432.

[40] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*. 452–461.

[41] Mónica Ribero, Jette Henderson, Sinead Williamson, and Haris Vikalo. 2022. Federating recommendations using differentially private prototypes. *Pattern Recognit.* 129 (2022), 108746.

[42] Yelong Shen and Ruoming Jin. 2012. Learning personal+ social latent factor model for social recommendation. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1303–1311.

[43] Ajit P. Singh and Geoffrey J. Gordon. 2008. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 650–658.

[44] Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, and Jian Tang. 2019. Session-based social recommendation via dynamic graph attention networks. In *Proceedings of the 12th ACM International Conference on Web Search and Data Mining*. 555–563.

[45] Jiliang Tang, Huiji Gao, Xia Hu, and Huan Liu. 2013. Exploiting homophily effect for trust prediction. In *Proceedings of the 6th ACM International Conference on Web Search and Data Mining*. ACM, 53–62.

[46] Jiliang Tang, Huiji Gao, and Huan Liu. 2012. mTrust: Discerning multi-faceted trust in a connected world. In *Proceedings of the 5th International Conference on Web Search and Web Data Mining*. ACM, 93–102.

[47] Jiliang Tang, Huiji Gao, Huan Liu, and Atish Das Sarma. 2012. eTrust: Understanding trust evolution in an online world. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 253–261.

[48] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the Advances in neural information processing systems*. 5998–6008.

[49] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. CoRR abs/1710.10903 (2017).

[50] Chen Wang, Yueqing Liang, Zhiwei Liu, Tao Zhang, and Philip S. Yu. 2021. Pre-training graph neural network for cross domain recommendation. In *CogMI*. IEEE, 140–145.

[51] 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.

[52] Xin Wang, Weike Pan, and Congfu Xu. 2014. Hgmf: Hierarchical group matrix factorization for collaborative recommendation. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*. 769–778.

[53] Chuhan Wu, Fangzhao Wu, Yang Cao, Yongfeng Huang, and Xing Xie. 2021. Fedgnn: Federated graph neural network for privacy-preserving recommendation. CoRR abs/2102.04925 (2021).

[54] 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. In *Proceedings of the IEEE Transactions on Knowledge and Data Engineering*.

[55] 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*. ACM, 235–244.

[56] Le Wu, Peijie Sun, Richang Hong, Yanjie Fu, Xiting Wang, and Meng Wang. 2018. SocialGCN: An efficient graph convolutional network based model for social recommendation. CoRR abs/1811.02815 (2018).

[57] Le Wu, Peijie Sun, Richang Hong, Yong Ge, and Meng Wang. 2021. Collaborative neural social recommendation. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 51, 1 (2021), 464–476.

[58] 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 *Proceedings of the World Wide Web Conference*. ACM, 2091–2102.

[59] Congying Xia, Caiming Xiong, S. Yu Philip, and Richard Socher. 2020. Composed variational natural language generation for few-shot intents. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*. 3379–3388.

[60] Chengxu Yang, QiPeng Wang, Mengwei Xu, Shangguang Wang, Kaigui Bian, and Xuanzhe Liu. 2020. Heterogeneity-aware federated learning. CoRR abs/2006.06983 (2020).

[61] Liangwei Yang, Zhiwei Liu, Yingtong Dou, Jing Ma, and Philip S. Yu. 2021. ConsisRec: Enhancing GNN for social recommendation via consistent neighbor aggregation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

[62] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. 2019. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology* 10, 2 (2019), 1–19.

[63] 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 SIGKDD*, Yike Guo and Faisal Farooq (Eds.), 974–983.

[64] Chuxu Zhang, Lu Yu, Yan Wang, Chirag Shah, and Xiangliang Zhang. 2017. Collaborative user network embedding for social recommender systems. In *Proceedings of the 2017 SIAM International Conference on Data Mining*, Nitesh V. Chawla and Wei Wang (Eds.), SIAM, 381–389.

[65] Yao Zhou, Haonan Wang, Jingrui He, and Haixun Wang. 2021. From intrinsic to counterfactual: On the explainability of contextualized recommender systems. arXiv:2110.14844. Retrieved from https://arxiv.org/abs/2110.14844.

[66] Yao Zhou, Jianpeng Xu, Jun Wu, Zeinab Taghavi, Evren Korpeoglu, Kannan Achan, and Jingrui He. 2021. PURE: Positive-unlabeled recommendation with generative adversarial network. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2409–2419.

Received April 2021; revised August 2021; accepted November 2021