axioms

MDPI

Article

FedGR: Federated Graph Neural Network for Recommendation Systems

Chuang Ma [1,†], Xin Ren [1,†], Guangxia Xu [2,*] and Bo He [1]

[1] School of Software Engineering, Chongqing University of Posts and Telecommunications, Chongqing 400065, China
[2] The Cyberspace Institute of Advanced Technology, Guangzhou University, Guangzhou 510006, China
* Correspondence: xugx@gzhu.edu.cn
† These authors contributed equally to this work.

Abstract: Social recommendation systems based on the graph neural network (GNN) have received a lot of research-related attention recently because they can use social information to improve recommendation accuracy and because of the benefits derived from the excellent performance of the graph neural network in graphic data modeling. A large number of excellent studies in this area have been proposed one after another, but they all share a common requirement that the data should be centrally stored. In recent years, there have been growing concerns about data privacy. At the same time, the introduction of numerous stringent data protection regulations, represented by general data protection regulations (GDPR), has challenged the recommendation models with conventional centralized data storage. For the above reasons, we have designed a flexible model of recommendation algorithms for social scenarios based on federated learning. We call it the federated graph neural network for recommendation systems (FedGR). Previous related work in this area has only considered GNN, social networks, and federated learning separately. Our work is the first to consider all three together, and we have carried out a detailed design for each part. In FedGR, we used the graph attention network to assist in modeling the implicit vector representation learned by users from social relationship graphs and historical item graphs. In order to protect data privacy, we used FedGR flexible data privacy protection by incorporating traditional cryptography encryption techniques with the proposed "noise injection" strategy, which enables FedGR to ensure data privacy while minimizing the loss of recommended performance. We also demonstrate a different learning paradigm for the recommendation model under federation. Our proposed work has been validated on two publicly available popular datasets. According to the experimental results, FedGR has decreased MAE and RMSE compared with previous work, which proves its rationality and effectiveness.

Keywords: social recommendation; graph neural network; federated learning; privacy protection

Citation: Ma, C.; Ren, X.; Xu, G.; He, B. FedGR: Federated Graph Neural Network for Recommendation Systems. *Axioms* **2023**, *12*, 170. https://doi.org/10.3390/axioms12020170

Academic Editor: Harish Garg

Received: 28 December 2022
Revised: 20 January 2023
Accepted: 23 January 2023
Published: 7 February 2023

1. Introduction

Recommendation systems have gradually attracted more and more attention in recent years [1,2]. When both are parties involved in recommendation systems, users and businesses can benefit. Graph neural networks (GNNs) have gradually become common in various fields due to their excellent performance in graph-data processing [3]. Therefore, the use of GNNs techniques in recommendation systems has become popular, and the corresponding recommendation problem has been transformed into a link-prediction problem in GNNs [4]. Numerous recent works have started to incorporate the social information of target users into recommendation model construction [2,5,6]. Thus, the low-dimensional representations of the surrounding neighbors are normally either averaged or fused with the low-dimensional representations of the target user's historical behavioral data using the attention mechanism to obtain a final more accurate low-dimensional representation of the user. Although the recommendation performance of recommendation systems has

been nicely improved by combining graph neural networks and user social network information [7], they all face a common problem: that data needs to be stored centrally [8,9]. Moreover, centralized storage has drawn public attention to data privacy and security issues. In addition, the recent introduction of a series of data privacy regulations, represented by the GDPR (https://gdpr-info.eu/ (accessed on 1 April 2022)), not only reflects the strict attitude of public institutions towards data privacy issues but also indicates that data privacy security should be an essential consideration in the construction of recommendation systems. Distributed data storage, which corresponds to centralized storage, has the property of data protection at the physical level, but it is also prone to data-island problems. Google formally proposed the concept of federated learning in 2016 in an attempt to find a balance between privacy protection and efficient use of data. However, they all suffer from these common drawbacks as follows. (1) In order to ensure the data security of users, traditional cryptographic technologies such as local differential privacy (LDP) [10] and homomorphic encryption (HE) [11] are generally introduced into federated learning to protect user data. However, some proof of work traditional differential privacy may not fit into the federation learning framework at all [12]. By summarizing these previous works, we propose the innovative work of federated graph neural network for recommendation systems(FedGR). It has effectively addressed the aforementioned challenges. First of all, in order to solve the discomfort of traditional cryptography in federated learning and the huge decline in recommendation performance, we have adopted two privacy protection methods in FedGR, " encryption decryption" and " noise-injection". By combining these two means, the degradation of recommendation performance can be controlled to the greatest extent and lead to excellent data privacy protection capabilities. Second, to reduce the load on the edge nodes and the communication load during model aggregation, we adopt a split model design where the item model is placed at the server side and the user model is left at the edge nodes. The benefits of this approach are diverse. Finally, we introduce the corresponding feature information for each item in the item model. We tested our proposed FedGR on two real datasets and showed significant improvements in both MAE and RMSE compared to some past federated recommendation work. The main contributions of our work are as follows: we are the first to apply the split-model approach to social recommendation in a federated learning framework and propose a novel data protection approach in a federated learning framework. The efficient combination of traditional cryptographic techniques and joint learning with the burden of complex models on edge nodes is addressed. Our content is organized as follows. We present our related work in Section 2, and then we elaborate on the details of our proposed work in Section 3. In Section 4, we will validate the effectiveness of our work on the Ciao and Epinions datasets, and finally we will conclude the paper and present directions for our future research work.

2. Related Work

In this section, we present the three areas most relevant to our work, namely, (1) social recommendation, (2) the graph neural network for recommendation systems, and (3) privacy-protection recommendation.

2.1. Social Recommendation

People in the same social circle tend to have similar interests [13], and they share their interests with each other; therefore, social networks are an essential source of improving the accuracy of recommendation systems. We classify social recommendation into two broad categories: traditional methods based on matrix factorization and deep learning models based on GNNs. Prior to the popularity of GNNs, researchers mainly used social information as a regularization term to constrain the final user representation and enrich the single-user representation. SoRec [14] proposed a cofactor decomposition approach that shares a common potential user feature matrix decomposed by user scores and social relations. SocialMF [15] considered that the behavior of user u is influenced by its direct

neighbors N_u. Therefore, the author takes the weighted average of the potential eigenvector of user u direct neighbor as the potential eigenvector estimate of user u. Following the emergence of GNNs, a large body of work has demonstrated their efficiency in social recommendation. Graphrec [2] and Graphrec+ [16] are from the same team. In both works, the authors use GNNs to learn user embeddings and item embeddings from social relationship graphs and historical item graphs and then pass these two embeddings through a multilayer perceptron to predict the final ratings. DiffNet [5] and DiffNet++ [17] use GNNs to model users' social relationships and interactive items, arguing that users' interests are diffused in the network, and the central user's propensity to consume is influenced by both low-order and high-order users to further improve the recommendations accuracy. Figure 1 below is a schematic diagram of social recommendation.

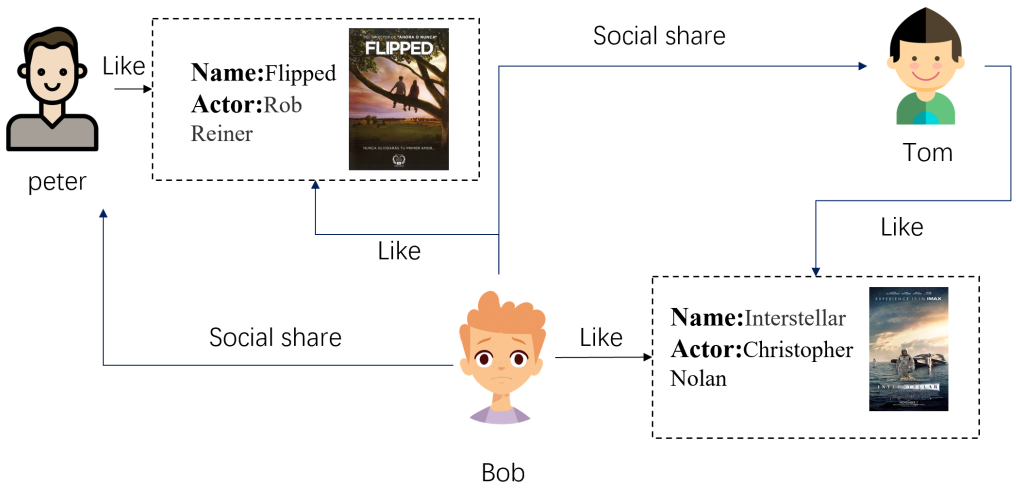

Figure 1. Social recommendation diagram.

2.2. Graph Neural Network for Recommendation Systems

Applications of GNNs in recommendation systems can be divided into general recommendation and sequential recommendation, where the former is static and does not incorporate temporal information. The latter is dynamical. Therefore, we will also discuss the application of GNNs in recommender systems in two aspects. Figure 2 shows the two most salient relations of GNN in the recommendation systems.

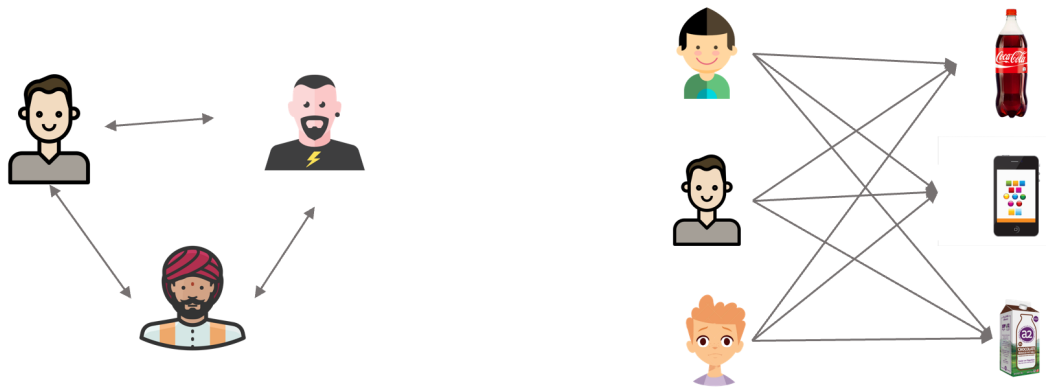

Figure 2. Recommend common graph structures in the system. user–user and user–item graphs.

2.2.1. For General Recommendation

GNNs can enhance user and item representation learning by explicitly encoding synergistic signals through node aggregation, which is more flexible and convenient for modeling multi-hop information than other models. GC-MC [18] uses GCN [19] as an encoder to learn user embedding and item embedding from user–item bipartite graph of

user items and is used to predict and complete the missing values in the scoring matrix. PinSAGE [20] is a recall algorithm proposed by Pinterest based on GraphSAGE [21], which learns aggregation functions directly instead of fixed nodes, thus differing from Transductive learning methods such as GCN. This is more in line with the changing needs of graph nodes in realistic situations.

2.2.2. For Sequential Recommendation

Converting sequential data into a sequential graph allows for more flexible primitive transformations for item selection, and GNNs can capture complex user interest preferences implicit in sequential behavior through a ring structure. HetGNN [22] constructs an edge between two consecutive item items with the same sequence as the edge type using all of the user's behavior sequences. FGNN [1] uses GRU [23] with an attention mechanism to iteratively update user preferences with item representation of sequence concept. A large body of past work has shown that GNN-based models have shown strong advantages in the recommendation domain. Therefore, in our work, we also adopt GNNs applied to user-model learning, and we mainly use GAT [24] .

2.3. Privacy-Protection Recommendation

As one of the early representative works in federation recommendation, in FCF [25], each edge user trains a local model using their own historical data, user embedding is updated locally, and finally only item embedding gradient data is uploaded to the server, but the gradient information may still leak some sensitive user data, so FedMF [26] authors additional use homomorphic encryption technique to protect the uploaded gradient information. As an early exploration of federated learning in the recommendation domain, the above two methods effectively protect the user embeddings and item embeddings, but they do not protect the interaction information between using items and learn some higher-order information. As a representative of federated learning in the field of recommendation systems in recent years, FedGNN [9] has demonstrated a fresh design idea. In order to protect data privacy, it has introduced two innovative technologies, namely, "Local Differential Privacy (LDP)" and "pseudo item labeling", on the framework of federated learning to protect user data. Unfortunately, FedGNN does not consider social information, which is an effective auxiliary to learn information in the learning process of low-dimensional representations, and its user model and item model are located at the edges, which imposes a large burden on model aggregation and local storage. By thoroughly considering the advantages and disadvantages of the above work, we proposed FedGR, which effectively combines GNNs, federated learning, and privacy protection technology and has considerably improved the recommendation performance.

3. Proposed Framework

In this section, we detail the details of our proposed FedGR framework. In the following, we will first introduce the notation and its associated conceptual definitions, as shown in Table 1.

Table 1. Symbols and definitions.

Symbol	Definitions and Descriptions
p_i	embedding of user i
q_j	embedding of item j
fe_i	friends embedding of user
r_{ij}	user i's rating score for item j, r_{ij} in [0,1,2,3,4,5]
α_{ij}	weighting factor of item j to user i
β_{ij}	weighting factor of friend j and user i
v_i	product of the item embeddings of user i and the corresponding score embeddings
h_i	collection of historical interaction item IDs for user i

Table 1. *Cont.*

Symbol	Definitions and Descriptions
h_i^{noise}	collection of noise item IDs added by user i
F_i	a set of features of item i,$F_i = f_1, f_2 \cdots f_n$
FE_i	a set of features embedding representation of item j,$FE_i = fe_1, fe_2 \cdots fe_n$
ψ_i	embedding representation learned by user i in the user–item graph
ψ_i^f	the embedding representation learned by user i friends in the user–item graph
μ_i	embedding representation learned by user i in the social graph
G_i^I	history item graph of user i
G_i^s	social relation graph of user i
G_i^f	friends item graph of user i
E_i^h	history item embedding set of user i
E_i^f	friends history item embedding set of uer i
e_{ij}	user i's rating of item j embedding
Δ_i	user i obtains the set of item IDs from the server

3.1. Model Overview

In this subsection, we present the overall architecture of FedGR and the overall model architecture is shown in Figure 3. In our FedGR, we jointly work with multiple edge servers to train a unified recommendation model. The whole implementation is as follows: Step 1, the server will randomly initialize a user model and send it to each edge server after encryption processing; step 2, each edge node will retrieve the local database to obtain the current user's historical interactive item sequence and send it to the server to obtain the embedding vector of the corresponding item after "noise injection" and "encryption decryption" processing; step 3, the local user model will be trained for specified rounds; step 4, the local model will be uploaded to the server and aggregated to form a unified global model through FedAVG [27] algorithm; and step 5, the server distributes the aggregated model to the edge nodes. Repeat steps 2–5 above until the number of training rounds is specified.

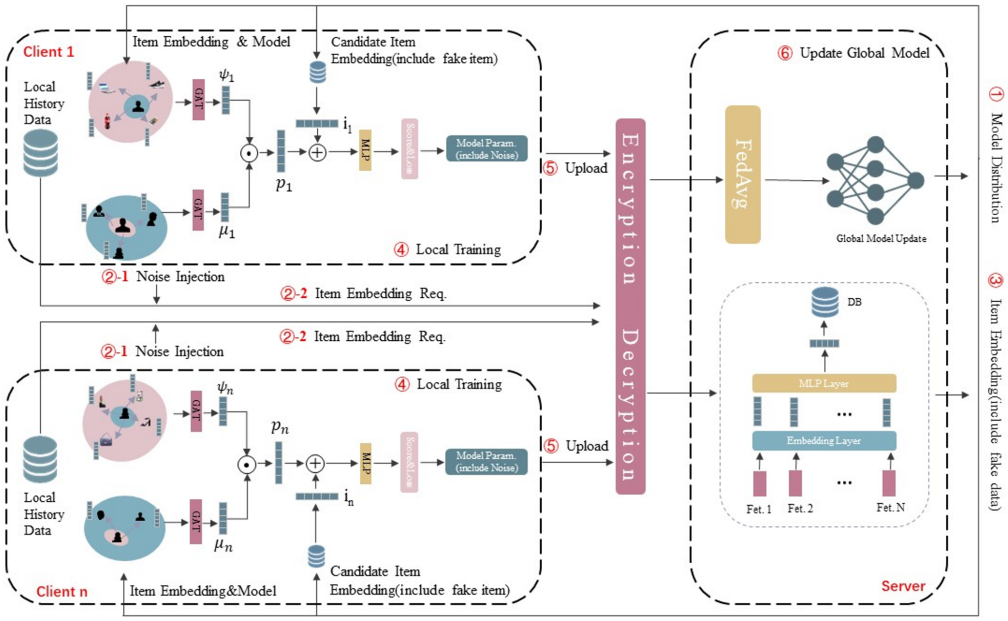

Figure 3. Overall overview of FedGR. The two dashed boxes on the left represent edge clients and the dashed boxes on the right represent server nodes. They are mainly responsible for training the fusion of item embeddings and model parameters, with the Encryption and Decryption module in between.

3.2. Item Model

In this section, we explain the learning process of item embeddings. Denote the set of features of item i. In our work, we incorporate the features of item i into the learning of hidden vectors for item representation. This enables better learning of individualized features among different item vectors. In FedGR, our item model employs a simple multilayer perceptron network that can be flexibly replaced for different application scenarios. The specific algorithmic steps are shown in Algorithm 1.

Algorithm 1 Item Embedding

Input: feature set of item i

$$F_i = f_1, f_2 \cdots f_n$$

Output: representation of item q_i

1: // embedding representation of each feature.
2: **for** k in F_i **do**
3: $\quad MLP_{embedding}(k) \rightarrow FE_i^k$
4: **end for**
5: // Connect all feature vectors and do feature crossing through multi-layer fully connected network .
6: $CONCAT(FE_i) \rightarrow mid\_var$
7: $MLP(mid\_var) \rightarrow q_i$
8: //the final embedding representation of item i .
9: **return** q_i

The input is a series of features of item i. Step 1 loops through all of the features of the input item. For item i, the initial state of each feature is represented by numbers. First, each feature is vectorized through the embedding layer on the right side of Figure 3, and all features are converted from numerical representation to low-dimensional vector representation. This prevents the sparsity problem caused by the lack of features of some items. The effective feature information crossover can be performed later . Step 2: After the embedding layer, each feature forms an embedding vector of uniform length. Through the multi-layer perceptron (MLP), each feature vector is effectively crossed through the MLP network to learn useful information from each other. Step 3: The final low-dimensional vector representation of item i is obtained through the transformation of vectors.

3.3. User Model

The main function of the user model is to learn a vector representation of the user by combining the historical behavioral data of the user with the contact information of the user's social friends. First, we will build the user–item graph and user–user graph based on the local history and social information of the users. We use ψ_i,μ_i, which represents the embedded representation that user i learned from the item graph and the embedded representation that user i learned from the social graph, respectively. These two vectors play a role in the end-user embedding representation construction from different perspectives, and we combine the learned representations from the two graph spaces. Finally, p_i represents the vector representation of user i. In the following subsections, we present the details of each module and the corresponding implementation algorithms.

3.3.1. Item Graph Representation

The user–item graph contains information not only about the items the user has historically interacted with but also about the user's attitude. Here, r_{ij} , as a real value representation, can reflect whether the user i has a positive or negative attitude towards the historical interactive item j. In the data set, we use r_{ij}, adopting the "five-point scale", which is not limited. Algorithm 2 gives us an ensemble learning example of user embeddings.

Algorithm 2 Item Graph Representation

Input: Input of the user–item graph and the embedding representation of the corresponding item (obtained from the server side)

$$G_i^I, E_i^h$$

Output: user–item embedding representation ψ_i

1: //Represent all scoring data as a low-dimensional vector .
2: $MLP_{embedding}(r_{ij}) \rightarrow e_{ij}$
3: //obtain user–item embedding(every item)
4: $CONCAT(E_i^h, e_i) \rightarrow mid\_var$
5: $MLP(mid\_var) \rightarrow v_i$
6: // Calculate the attention weight coefficient of items to users.
7: $CONCAT(v_i, embedding_i) \rightarrow mid\_var$
8: $MLP(mid\_var) \rightarrow \alpha_i$
9: // the finally user–item embedding representation
10: $MLP(\alpha_i \odot v_i) \rightarrow \psi_i$.
11: **return** ψ_i

Algorithm 2 obviously presents the detailed process of learning the user's representation ψ_i. Step 0: The user–item graph and item embeddings are fed into the algorithm as initial information. In addition, we inject a lot of noise data through "noise injection" to protect the user's real consumption data. The default score for these noisy data is zero. We will first set r_{ij} converted to low-dimensional representation e_{ij}, and we use e_i, which represents all rating-embedding sets of user i. Since our item embeddings are pre-trained on the server side, we can directly fuse the original embedding representation of the item with the current user-rated embedding of the item. Use the following Equation (1) as follows:

$$CONCAT(E_i^h, e_i) \tag{1}$$

where E_i^h represents the embedded set representation of user i historical items, and e_i represents the embedding set of user i ratings on the corresponding items. There are two ways to aggregate the historical interaction items of all users. One is to use simple mean values as weight coefficients in the aggregation process. The alternative is to use the attention mechanism for aggregation to differentiate the aggregation weight of each item. Indeed, the latter is better. The attention network is defined as Equation (2):

$$\alpha_{ij} = W_2 \cdot \sigma(W_1 \cdot CONCAT(v_{ij}, embedding_i) + b_1) + b_2 \tag{2}$$

where v_{ij} represents the unified representation of the user i rating embedding of item j and the initial embedding of item j, and $embedding_i$ represents the initial embedding of user i. Item aggregation use follows Equation (3) representation:

$$\psi_i = MLP(\alpha_i \odot v_i) \tag{3}$$

v_i denotes the uniform representation of all items embedding and corresponding rating embedding in the item graph of user i, $\odot$ represents the point between vectors, and α_i denotes the set of all corresponding weight coefficients.

3.3.2. Social Aggregation

Similar to previous works that learn embedding representations from user–item pairs, we also introduce an attention mechanism in the social relationship aggregation process, where the influence of different friends is different and reflected by different attention weighting coefficients. The exact procedure is shown in Algorithm 3.

Algorithm 3 Social Representation

Input: input social graph/user–friend–item graph/friends item-embedding set

$$G_i^s, G_i^f, E_i^f$$

Output: user i social embedding representation μ_i

1: // calc the friends j user–item embedding representation, use Algorithm 2
2: $MLP(\alpha_j \odot v_j) \rightarrow \psi_j$
3: // calc the user–user attention score
4: $MLP(CONCAT(embedding_i, \psi_j)) \rightarrow \beta_{ij}$
5: //cacl the user i social embedding representation use all friends j and attention score
6: $MLP(\beta_i \odot \psi_i^f) \rightarrow \mu_i$
7: **return** μ_i

The embeddings corresponding to the user social graph, friend history item interaction graph, and friend history consumption item are fed into the model as raw data. Similar to Algorithm 2, since the initial scores of all friends on items are numerical scores from 1 to 5, they need to be represented as low-dimensional vectors first. The item embeddings and score embeddings can then be fused through a multi-layer neural network. The specific formula is as follows Equation (4):

$$MLP(CONCAT(E_i^f, e_i)) \rightarrow v_i \tag{4}$$

Here, E_i^f denotes the embedding representation of all first-order friends of user i that have consumed items, and e_i is the corresponding rating embedding. The computation of the attention scores between friends and consumed items is similar to the one in Equation (2), but the attention scores of first-order friends and their history items should be computed first before the user and user attention scores are calculated. The calculation formula of the user–user attention score is shown in step 5 of Algorithm 3, where $embedding_i$ represents the initial embedding representation of the current user to be calculated and ψ_i^f represent the embedded representation set learned by all friends of user i from their corresponding historical product graph. The final β_i is the set of weight coefficients of user i's friends. Through algorithm 3, we can obtain the embedding representation of user i in the social relationship graph by combining the influence of different first-order friends as follows in Equation (5).

$$\mu_i = MLP(\beta_i \odot \psi_i^f) \tag{5}$$

Ultimately, user i learns the final low-dimensional vector representations from the user–item graph and user–user graph, respectively, ψ_i and μ_i,$\oplus$ representative vector splicing. Therefore, the final low-dimensional vector learned by user i from user model is characterized by the following Equation (6):

$$p_i = MLP(\psi_i \oplus \mu_i) \tag{6}$$

3.4. Security Model

In our proposed work, we innovatively adopt two approaches to secure user data privacy. We can flexibly adjust the data privacy protection strength and optimize the recommendation performance for different recommendation scenarios. First, in step 2 as shown in Figure 3, the edge node obtains the embedding representation of the corresponding item from the server through the real personal history item ID, and first, the "Noise Injection" operation will be performed locally. The relevant formulation is as follows: In our proposed work, we innovatively adopt two approaches to secure user data privacy. The relevant formula is shown as follows in Equations (7) and (8):

$$\Delta_i = h_i + h_i^{noise} \tag{7}$$

$$len(h_i^{noise}) = \epsilon \cdot len(h_i) \tag{8}$$

In the above formula, h_i represents the item ID set that user i has genuinely consumed, and h_i^{noise} represents the items that user i has not consumed. It is the forged data added by "noise injection". Together, they form the ID set Δ_i. This can effectively confuse the "thief" to speculate about the real consumption habits and relevant data of user i. In Function (8), the hyperparameter ϵ is used to control the proportion of "noisy data" added. The greater the level of privacy protection, the better, but at the same time, the performance drops. Through our experiments, we observed that when ϵ, a favorable balance between privacy protection and recommendation performance can be achieved when $\epsilon \in (0.2, 0.35)$. After the noise injection process, the second security protection can be entered:module in our framework "encryption-decryption" module. The specific implementation is shown in Figure 4.

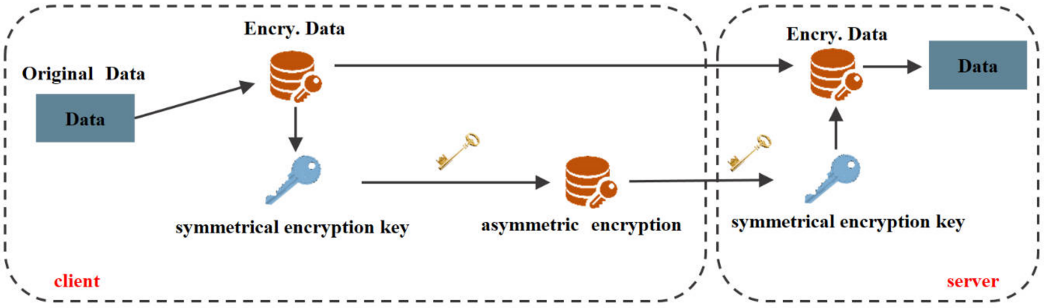

Figure 4. Encryption protection module. The target data will be encrypted symmetrically first, followed by an asymmetric encryption protection module.

The data set Δ_i, which is processed by the previous "noise injection", is firstly encrypted by a symmetrical encryption model; it is particularly suitable for encrypting a large number of model parameters and can effectively protect large files. In FedGR, we use it to encrypt the original data, which will eventually generate two parts: the encrypted data containing the original data and the symmetric encryption key. However, since symmetric encryption uses the same key in both the encryption and decryption phases, there is a risk of losing the secret key. Thus, after the file is symmetrically encrypted, asymmetric encryption is performed on the symmetric encryption secret key, which is signed using the private key stored locally by the edge server. At this point, the files transmitted to the server include noisy data with symmetric encryption, and symmetric encryption keys with asymmetric encryption processing. After receiving these two, similar to the client requesting data from the server, when the server distributes model parameters and item-embedding to the client, it will also go through the "encryption-decryption" module. The difference is that "noise injection" will not be performed on the server. Next, the client puts the noisy item embeddings into the user model for training to obtain the final model parameters, which are clearly trained from the noisy data, and it is also difficult for the thief to intercept the model parameters to invert the corresponding user consumption behavior data.

4. Experiment

In this section, we will select several benchmark models from different perspectives and evaluate the performance of our proposed FedGR model on two real data sets.

4.1. Experimental Settings

4.1.1. Datasets

We chose two highly popular publicly available social datasets, Ciao and Epinions (http://www.cse.msu.edu/~tangjili/trust.html (accessed on 1 April 2022)), which are extremely commonly used in social recommendation scenarios and have been used for

performance evaluation in a large number of works in this domain. Specifically, what is crucial for our work is that both datasets contain categorical information, which we can incorporate as essential feature information in the term-representation learning process. The statistics of the two datasets are shown in Table 2.

Table 2. Statistics of the datasets.

Datasets	Ciao	Epinions
Users	2248	22,168
Items	16,862	296,277
Ratings	36,065	920,075
Social Connections	57,545	355,812
Rating Scale	[1,5]	[1,5]

4.1.2. Evaluation Metrics

We use two extremely popular evaluation metrics: MAE (mean absolute error) and RMSE (root mean square error), both of which are the most commonly used metrics to measure the accuracy of variables and mainly reflect the scale of deviation of the predicted value from the true value. Smaller values for both metrics represent better performance. The specific calculation formula of these two formulas is shown Equations (9) and (10):

$$MAE(X,h) = \frac{1}{m}\sum_{i=1}^{m}|h(X^{(i)}) - y^{(i)}| \tag{9}$$

$$RMSE(X,h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(h(X^{(i)}) - y^{(i)})^2} \tag{10}$$

where $h(X^{(i)})$ and $y^{(i)}$, respectively, represent the predicted value and the real value, and m represents the total number of data.

4.1.3. Parameter Settings

In our tests, we adopt a modern way of dividing the datasets, which we call UBC (user-based cutting). In the past, data partitioning for federated learning works were essentially based on centralized training dataset cuts, but we argue that this is not realistic and that it is more reasonable to partition edge users based on features learned by federated learning. Suppose M represents the set of all edge user nodes, $M = m_t, m_v, m_e$, where m_t represents the training user set, m_v represents the validation user set, and m_e represents the test user set. In total, 80% of the user set is divided into the training set, 10% into the validation set, and 10% into the test set. In all experiments, we initialize the parameters based on Gaussian distributions. We also control the number of first-order users, mainly to prevent some edge users from having good social ties and a much higher than average number of first-order friends, while others only have a small number of first-order friends or even no social relation, leading to biased learning results, so we set the number of friends to $5, 10, 15, 20$. The embedding size d is tuned from $8, 16, 32, 64$; the noise ratio in FedGR is chosen as $0, 0.1, 0.2, 0.3, 0.4, 0.5$; and for the comparison method with local differential privacy protection, we set a gradient clipping threshold of 0.3, a laplace noise length of 0.1, a learning rate chosen as $0.1, 0.05, 0.01$, and the number of local edge users training before each model training. Finally, the training stopping criterion is to reach the pre-defined number of training rounds.

4.1.4. Baselines

In order to evaluate our proposed framework more comprehensively, we have selected three groups of methods to compare with the FedGR method, including the traditional social recommendation systems model (SoReg, SocialMF), the recommendation model combined with deep graph neural network technology (GraphRec, GCMC+SN), and the recommendation systems model with privacy (FeSoG, FedMF).

SoReg [28]: A factor analysis recommendation algorithm based on the probability matrix decomposition.
SocialMF [29]: Introducing trust propagation in matrix decomposition, the user indicates that friends close to that user indicate.
GraphRec [2]: Graph neural networks are used to learn user embeddings and item embeddings from user history product graphs and social graphs.
GCMC+SN [25]: A graph-neural-network-based recommendation model is used to generate embeddings for each user in the social network using the node2vec technique.
FeSoG [30]: A social recommendation system with privacy protection, using local differential privacy (LDP) and pseudo-item labeling as a means of user data privacy protection.
FedMF [26]: The representation of each user is computed by matrix factorization, and homomorphic encryption is used to protect the user data from disclosure.

4.2. Quantitative Results

The performance of all compared models is shown in Table 3, and for a more visual observation, we generate Figure 5.

Table 3. Empirical results compared with different baseline methods. The last row will show the percentage improvement of FedGR over the remaining federally recommended approaches.

Method	Ciao MAE	Ciao RMSE	Epinions MAE	Epinions RMSE
SoReg	0.8627	1.1021	0.9119	1.1703
SocialMF	0.8270	1.0501	0.8837	1.1328
GraphRec	0.8141	1.0133	0.8326	1.0814
SoRGCMC+SN	0.7824	1.0031	0.8480	1.1070
FeSoG	1.4937	1.9136	1.3847	1.7969
FedMF	2.0792	2.4216	1.5254	2.0685
FedGR	1.3650	1.7941	1.2773	1.5806
Improvement (%)	**8.6**	**6.2**	**7.7**	**12.1**

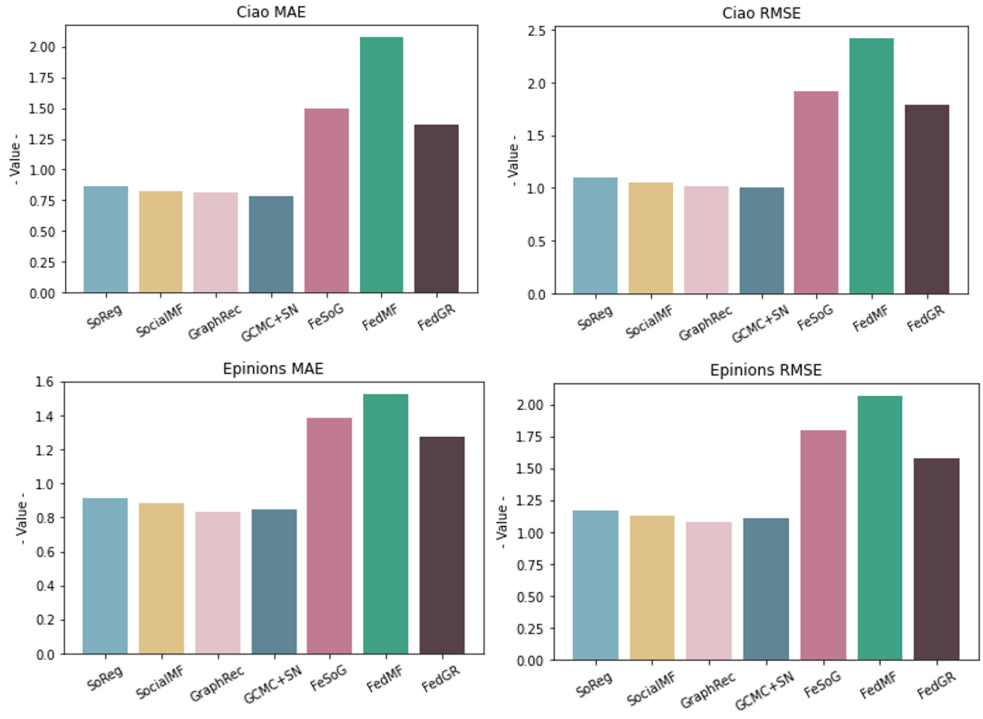

Figure 5. Ciao Datasets: MAE, RMSE Epinions datasets: MAE and RMSE. The vertical axis indicates the performance parameters of each algorithm in the corresponding dataset. The smaller the value, the better the performance.

We have the following observations from the data in the results. (1) Although both are based on social information for recommendation prediction, it is obvious that the GraphRec, GCMC+SN model combined with deep neural network ultimately performs better on both datasets than SoReg and SocialMF using traditional matrix decomposition techniques. (2) With the same dataset, all recommendation models based on the federated learning approach perform worse than the centralized framework processing. (3) The models that incorporate the graph neural network technology all perform better than those that do not incorporate the graph neural network technology. (4) Among all of the federated recommendation methods, our proposed FedGR has the best performance. The main reasons for this are as follows: firstly, we add item category information as a feature to the item-embedding learning process, which enriches the item-embedding representation. Second, we add a graph neural network to the user model to construct the social graph and item graph of the user. We effectively learn the hidden vector representations of the users. Thirdly, our proposed two privacy protection methods can minimize the degradation of recommendation performance under the premise of privacy protection, especially our "Encryption-Decryption" method, which fundamentally does not produce the degradation of recommendation performance.

4.3. Analysis of Parameters

In this section, we analyze some of the main hyperparameters in FedGR that determine the performance of FedGR, including (a) the number of friends in the social relation grap; (b) the number of items in the item graph; (c) the number of "noise items" into the user–item graph construction; for each parameter with different values, we mainly use the MSE and RMSE values as measures. The results are shown in Figure 6.

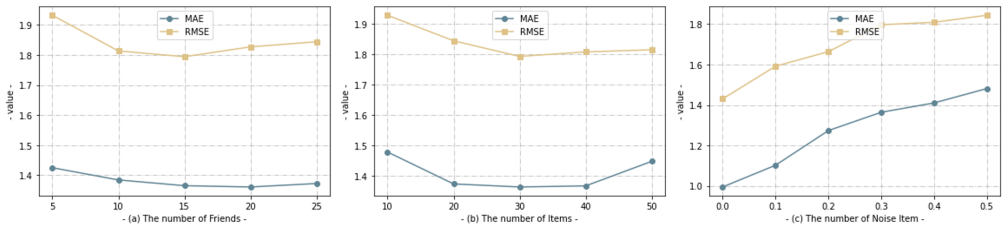

Figure 6. The performance trend of FedGR under different parameters, by observing MAE and RMSE.

As shown in (a), the number of first-order friends has a strong influence on the final MAE/RMSE when the social graph is constructed, in FedGR; initially, when the number of users is less than 15, the MAE/RMSE all show a rapid decreasing trend, mainly because as the number of friends increases, it can provide more learning for the user representations' more valid information for the learning of user representations. However, when the number of friends is larger than 15, the decline starts to slow down, and even after 25, both MAE/RMSE show a slight rebound trend, although the increase in rebound varies. We believe that the reasons for this are manifold and can be summarized into two main aspects: (1) there are a large number of users whose first-order number of friends is originally not 15; and (2) too many friends will introduce noisy data that interfere with the target user's prediction and reduce the overall performance of the final model. By looking at the figure shown in (b), we can see that a similar situation to that in figure (a) arises. Figure (c) shows an entirely different dynamic from (a,b), where MAE/RMSE both increase with the increase in noise ratio. We found that in FedGR, if the noise item is set to 0, the performance is even better than some proposed social recommendation models in the past, but as the number of noise items increases, the MSE/RMSE both tend to rise rapidly, so knowing how to minimize the performance loss in social recommendation while protecting data privacy and security has been a key concern in this area.

5. Conclusions and Future Work

In this paper, we innovatively propose a federated learning framework based on a split-model social recommendation framework, which we call FedGR. To the best of our

knowledge, our work is the first social recommendation systems model that decouples the user and item models and is privacy-protected. FedGR brings the advantages of GNN and social information to the recommendation systems and brings great accuracy to them. At the same time, in response to recent privacy protection concerns, our work builds on previous frontier work in federated learning. The combination of the two makes FedGR exhibit excellent performance. In FedGR, we achieve a excellent trade-off between privacy protection and recommendation performance while providing users with a lot of flexibility. To improve the recommendation performance, we incorporate the feature type information of items when retraining the item embeddings, and we propose two privacy preserving methods to improve privacy protection and minimize the loss in the recommendation performance. Finally, we compare our proposed FedGR with several different reference types on several real public datasets and show the effectiveness of our work on all datasets. Although, FedGR has shown its validity, we believe that it still has the following problems. (1) It does not consider the time series information of users' consumption. In our work, we purely consider the types of items consumed by users in the past, but the reality is that users' interests alter over time, which has been well established in recent years by a large amount of work on sequential and session recommendations [31]. (2) In FedGR, we are still using the more traditional FedAvg model parameter aggregation approach, although there has been a lot of work in the past that has confirmed its effectiveness and is extremely popular. However, simply averaging over all model parameters as different model parameters may result in some loss of model performance. Recently, there has been a lot of work proposing an aggregation approach similar to the attention mechanism [32], where the final model parameters are obtained by summing different weight values according to the variability of each edge user. Therefore, our future work will consider introducing the recurrent neural network (RNN) or transformer and alternative technologies to explore the evolution of user interests, which are closer to the actual situation. Second, we will introduce a better federated learning algorithm to replace the FedAVG algorithm we currently use. Moreover, all of our work will still be conducted under the topic of socially recommended privacy protection.

Author Contributions: X.R. undertook this study and drafted the manuscript. B.H. sorted out the relevant literature and evaluated its applicability. G.X. and C.M. provided professional technical guidance on the implementation details of the article and summarized and reviewed the contents of the manuscript. All authors read and approved the final manuscript.

Funding: This work is supported by the National Natural Science Foundation of China (Grant No. 62272120, 62106030), the Technology Innovation and Application Development Projects of Chongqing (Grant No. cstc2021jscx-gksbX0032, cstc2021jscx-gksbX0029), the Research Program of Basic Research and Frontier Technology of Chongqing (Grant No. cstc2021jcyj-msxmX0530), and the Key R & D plan of Hainan Province (Grant No. ZDYF2021GXJS006).

Acknowledgments: The authors would like to thank the editor and anonymous reviewers for their valuable comments and suggestions on this paper.

Conflicts of Interest: Authors declare no conflict of interest.

References

1. Qiu, R.; Li, J.; Huang, Z.; Yin, H. Rethinking the item order in session-based recommendation with graph neural networks. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, Beijing, China, 3–7 November 2019; pp. 579–588.
2. Fan, W.; Ma, Y.; Li, Q.; He, Y.; Zhao, E.; Tang, J.; Yin, D. Graph neural networks for social recommendation. In Proceedings of the The World Wide Web Conference, San Francisco, CA, USA, 13–17 May 2019; pp. 417–426.
3. Wang, D.; Cui, P.; Zhu, W. Structural deep network embedding. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 1225–1234.
4. Nasiri, E.; Berahmand, K.; Li, Y. Robust graph regularization nonnegative matrix factorization for link prediction in attributed networks. *Multimed. Tools Appl.* **2023**, *82*, 3745–3768. [CrossRef]
5. Wu, L.; Sun, P.; Fu, Y.; Hong, R.; Wang, X.; Wang, M. A neural influence diffusion model for social recommendation. In Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, Paris, France, 21–25 July 2019; pp. 235–244.

6. Xu, G.; Wu, X.; Liu, J.; Liu, Y. A community detection method based on local optimization in social networks. *IEEE Netw.* **2020**, *34*, 42–48. [CrossRef]
7. Berahmand, K.; Mohammadi, M.; Saberi-Movahed, F.; Li, Y.; Xu, Y. Graph regularized nonnegative matrix factorization for community detection in attributed networks. *IEEE Trans. Netw. Sci. Eng.* **2023**, *10*, 372–385. [CrossRef]
8. Xu, G.; Dong, J.; Ma, C.; Liu, J.; Cliff, U.G.O. A Certificateless Signcryption Mechanism Based on Blockchain for Edge Computing. *IEEE Internet Things J.* **2022**. [CrossRef]
9. Wu, C.; Wu, F.; Cao, Y.; Huang, Y.; Xie, X. Fedgnn: Federated graph neural network for privacy-preserving recommendation. *arXiv* **2021**, arXiv:2102.04925.
10. Dwork, C. Differential privacy: A survey of results. In Proceedings of the International Conference on Theory and Applications of Models of Computation, Xi'an, China, 25–29 April 2008; pp. 1–19.
11. Gentry, C. Fully homomorphic encryption using ideal lattices. In Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing, Bethesda, MD, USA, 31 May–2 June 2009; pp. 169–178.
12. Hao, M.; Li, H.; Luo, X.; Xu, G.; Yang, H.; Liu, S. Efficient and privacy-enhanced federated learning for industrial artificial intelligence. *IEEE Trans. Ind. Inform.* **2019**, *16*, 6532–6542. [CrossRef]
13. Xu, G.; Li, W.; Liu, J. A social emotion classification approach using multi-model fusion. *Future Gener. Comput. Syst.* **2020**, *102*, 347–356. [CrossRef]
14. Ma, H.; Yang, H.; Lyu, M.R.; King, I. Sorec: Social recommendation using probabilistic matrix factorization. In Proceedings of the 17th ACM Conference on Information and Knowledge Management, Napa Valley, CA, USA, 26–30 October 2008; pp. 931–940.
15. Jamali, M.; Ester, M. A matrix factorization technique with trust propagation for recommendation in social networks. In Proceedings of the Fourth ACM Conference on Recommender Systems, Barcelona, Spain, 26–30 September 2010; pp. 135–142.
16. Fan, W.; Ma, Y.; Li, Q.; Wang, J.; Cai, G.; Tang, J.; Yin, D. A graph neural network framework for social recommendations. *IEEE Trans. Knowl. Data Eng.* **2020**, *34*, 2033–2047. [CrossRef]
17. Wu, L.; Li, J.; Sun, P.; Hong, R.; Ge, Y.; Wang, M. Diffnet++: A neural influence and interest diffusion network for social recommendation. *IEEE Trans. Knowl. Data Eng.* **2020**, *34*, 4753–4766. [CrossRef]
18. Berg, R.v.d.; Kipf, T.N.; Welling, M. Graph convolutional matrix completion. *arXiv* **2017**, arXiv:1706.02263.
19. Defferrard, M.; Bresson, X.; Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. *Adv. Neural Inf. Process. Syst.* **2016**, *29*.
20. Ying, R.; He, R.; Chen, K.; Eksombatchai, P.; Hamilton, W.L.; Leskovec, J. Graph convolutional neural networks for web-scale recommender systems. In Proceedings of the 24th ACM Sigkdd International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; pp. 974–983.
21. Hamilton, W.; Ying, Z.; Leskovec, J. Inductive representation learning on large graphs. *Adv. Neural Inf. Process. Syst.* **2017**, *30*.
22. Wang, W.; Zhang, W.; Liu, S.; Liu, Q.; Zhang, B.; Lin, L.; Zha, H. Beyond clicks: Modeling multi-relational item graph for session-based target behavior prediction. In Proceedings of the The Web Conference 2020, Taipei, Taiwan, 20–24 April 2020; pp. 3056–3062.
23. Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv* **2014**, arXiv:1406.1078.
24. Velickovic, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y. Graph attention networks. *Stat* **2017**, *1050*, 20.
25. Ammad-Ud-Din, M.; Ivannikova, E.; Khan, S.A.; Oyomno, W.; Fu, Q.; Tan, K.E.; Flanagan, A. Federated collaborative filtering for privacy-preserving personalized recommendation systems. *arXiv* **2019**, arXiv:1901.09888.
26. Chai, D.; Wang, L.; Chen, K.; Yang, Q. Secure federated matrix factorization. *IEEE Intell. Syst.* **2020**, *36*, 11–20. [CrossRef]
27. Mills, J.; Hu, J.; Min, G. Communication-efficient federated learning for wireless edge intelligence in IoT. *IEEE Internet Things J.* **2019**, *7*, 5986–5994. [CrossRef]
28. Ma, H.; Zhou, D.; Liu, C.; Lyu, M.R.; King, I. Recommender systems with social regularization. In Proceedings of the Fourth ACM International Conference on Web Search and Data Mining, Hong Kong, China, 9–12 February 2011; pp. 287–296.
29. Guo, H.; Tang, R.; Ye, Y.; Li, Z.; He, X. DeepFM: A factorization-machine based neural network for CTR prediction. *arXiv* **2017**, arXiv:1703.04247.
30. Liu, Z.; Yang, L.; Fan, Z.; Peng, H.; Yu, P.S. Federated social recommendation with graph neural network. *ACM Trans. Intell. Syst. Technol. (TIST)* **2022**, *13*, 1–24. [CrossRef]
31. Wu, S.; Sun, F.; Zhang, W.; Xie, X.; Cui, B. Graph neural networks in recommender systems: A survey. *ACM Comput. Surv.* **2022**, *55*, 1–37. [CrossRef]
32. Reddi, S.; Charles, Z.; Zaheer, M.; Garrett, Z.; Rush, K.; Konečnỳ, J.; Kumar, S.; McMahan, H.B. Adaptive federated optimization. *arXiv* **2020**, arXiv:2003.00295.