TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214
DOI: 10.26599/TST.2026.9010005

Federated Heterogeneous Graph Contrastive Learning for Privacy-preserving Recommendation

Yuan Wang, Yu Wang and Yiwen Zhang*

School of Computer Science and Technology, Anhui University, Hefei 230093, China
Emails: yuanwang@stu.ahu.edu.cn, wangyuahu@stu.ahu.edu.cn, zhangyiwen@ahu.edu.cn

Abstract: Recommender systems aim to predict users' interests and needs by analyzing their historical interaction data, thereby providing personalized content and product suggestions. Traditional recommendation methods, such as collaborative filtering and hybrid systems, achieve significant success in improving user experience and driving sales but fall short in handling user privacy issues. Federated Recommendation (FedRec) emerges to address these limitations, integrating the principles of federated learning (FL), a distributed learning approach that allows multiple clients to collaboratively train models without sharing their personal data. Despite FedRec making significant progress in protecting user privacy, it also faces some performance and personalization challenges. i) **the model performance deficiencies** caused by limitations in data availability; ii) **the heterogeneity** caused by uneven distribution of client data. In this paper, we propose a novel framework, named **Fed**erated **H**eterogeneous **G**raph **C**ontrastive **L**earning (FedHGCL), which utilizes heterogeneous information to construct multiple augmented views for contrastive learning (CL) to enhance FedRec. Firstly, we introduce **federated heterogeneous graph contrastive learning**, where each user locally constructs a small CL-based multi-view framework to enhance recommendation performance. Secondly, we design a **user sampling** strategy for **data augmentation (DA)** on the client-side and model updates on the server to balance the training data for each user. Last but not least, we prove that the CL and DA used in FedHGCL meet the requirements for privacy-preserving recommendation. Extensive experiments on three real-world datasets prove the effectiveness of FedHGCL in personalized recommendation and privacy protection.

Key words: Federated Recommendation, Contrastive Learning, Heterogeneous Graph Neural Network, Privacy-preserving Recommendation

1 Introduction

Recommender systems [1–3] are crucial in managing the information overload in people's daily lives, such as through news push [4,5] and shopping suggestions [6,7]. Collaborative Filtering (CF) [8–10] is a critical task in recommender systems, employing historical user-item bipartite graph to deduce vector representations for users and items. For example, Matrix Factorization (MF) [11–13] is a significant technique in CF. This technique represents users and items as latent feature vectors, predicting interactions between users and items through linear combinations of these vectors. However, the collection and utilization of user data by conventional recommender systems have significantly heightened concerns regarding user privacy. In response

• Yuan Wang, Yu Wang, and Yiwen Zhang are with the School of Computer Science and Technology, Anhui University, Hefei 230093, China. E-mails: yuanwang@stu.ahu.edu.cn; wangyuahu@stu.ahu.edu.cn; zhangyiwen@ahu.edu.cn.
∗ To whom correspondence should be addressed.
Manuscript received: 2025-10-27; revised: 2025-11-24; accepted: 2025-12-29.

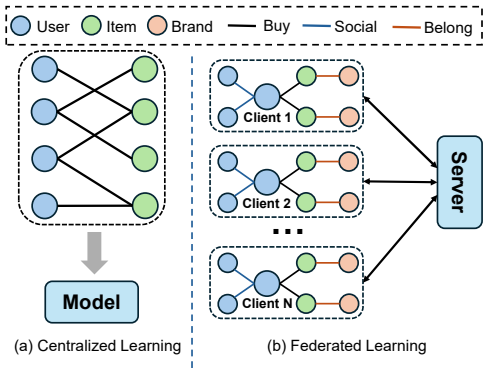


Fig. 1 Centralized learning trains all user interaction data on a central server, including heterogeneous information not displayed. Federated learning maintains user information locally, only uploading and requesting non-sensitive data to the server. Heterogeneous information encompasses, but is not limited to, users' social relationships, historical interactions, and the categories of items. Best view in color.

to these issues, the United States has implemented regulations such as the General Data Protection Regulation (GDPR)* and the California Consumer Privacy Act (CCPA)† to safeguard user privacy.

Federated Learning (FL) [14,15] is a distributed machine learning framework that allows multiple participants to collaboratively train models without direct data exchange, thus facilitating data analysis and modeling with privacy protection. Federated Recommendation (FedRec) [8,16,17], the integration of FL and recommender systems, demonstrates the privacy protection benefits of FL while enhancing the personalization of recommendation services. As shown in Figure 1, in FedRec, the users' interaction data (such as clicks, purchases, ratings, etc.) remain on the users' devices. This approach facilitates protecting user privacy and reduces the risk of data leakage [12,18]. Clients train parts of the model using their local data (usually the model's parameters or gradients) and then send updates of these parameters or gradients to the central server. This paradigm is applied in various fields, such as general recommendation-based [12,17], social recommendation-based [19,20], and heterogeneous graph-based [21,22] recommendation scenarios. Moreover, traditional recommendation algorithms are adapted to federated environments, including methods like collaborative filtering [8,12], graph neural networks [17,23], and self-supervised learning [18,20]. However, current FedRec still presents several urgent limitations to be addressed. In particular, most existing FedRec methods primarily operate on homogeneous or social graphs and lack explicit modeling of heterogeneous relational semantics, while heterogeneous graph contrastive learning (HGCL [2,24,25]) methods are typically designed for centralized settings and assume full access to user–item interactions.

CH1: Model Performance Deficiencies. The original intention of federated learning is to protect user privacy, but while it protects data privacy, it may also restrict data availability, possibly leading to inadequate model learning. For example, each client can only access local, limited user data, and uploading the results trained from these data to the server may cause model bias, thus reducing performance.

CH2: Heterogeneity and Personalization. In federated learning, participants' data typically exhibit varied distributions. Such heterogeneity hampers the model's ability to effectively generalize the information learned from one client to others. For instance, some users may have rich interaction data, while others might have extremely sparse data, making it difficult to achieve unified optimization and enhancement on the server-side.

To address the aforementioned challenges, we propose **Fed**erated **H**eterogeneous **G**raph **C**ontrastive **L**earning (FedHGCL), which leverages the structural advantages of heterogeneous graphs (HGs) [26] and the mechanisms of contrastive learning (CL) [27] in a federated setting to enhance the model's representational capabilities. For **Challenge 1**, many studies [18,19,22] attribute performance declines to a lack of data on the client-side. Considering that HGs and CL are important methods for mitigating the sparsity in recommendation data, we endeavor to incorporate these approaches to further explore local hidden preferences and information to improve recommendation performance. Specifically, we first involve utilizing various meta-paths (from users to items) to obtain multiple semantic subgraphs, each often representing a specific relationship within heterogeneous information. Then, using initialized embeddings, we perform graph convolution [9,28] operations on such subgraphs to obtain user and item embeddings based on different semantics. Each subgraph is considered an enhanced view, where embeddings from various views are contrasted sequentially to extract rich supervised signals,

*https://gdpr-info.eu/
†https://oag.ca.gov/privacy/ccpa

alleviating performance issues due to data sparsity. Regarding **Challenge 2**, users' personalization is considered a promising solution to alleviate client heterogeneity [17,29]. Following this direction, we design a server-based global augmentation strategy to effectively augment users' existing heterogeneous data. First, we propose a user sampling method determined by the edge count in clients' HG, where clients with sparse edges are prioritized for updates and global enhancements to expand existing connections. This method averages the data volume per user, not only mitigating the generalization issues brought by heterogeneity but also considering the high communication costs. In a nutshell, global augmentation utilizes local differential privacy (LDP)-processed [30–32] user and item embeddings on the server to calculate similarities that augment new nodes and connections locally for users. **Our main contributions are summarized as follows:**

- We propose FedHGCL, which employs heterogeneous graphs and multi-view contrastive learning to effectively augment user supervised signals. This framework is used to overcome the performance challenges brought by the data sparsity on the client-side. Furthermore, we provide a rational analysis of the privacy protection process for FedHGCL.
- We design an update and data augmentation strategy based on user sampling, to balance client data and alleviate the generalization problem caused by heterogeneity. The sampling strategy further improves communication costs in FedRec.
- We conduct extensive experiments on three real datasets, and the results show that our method demonstrates significant advantages over the existing FedRec. Even compared to some centralized learning methods, FedHGCL achieves comparable or better results.

2 Related Work

In this section, we first introduce the basic federated recommendation, which is meaningful as a specific application scenario. Furthermore, contrastive learning and heterogeneous graph neural networks are the technologies used in the proposed FedHGCL. We adapt them for the federated recommendation scenario and introduce a novel perspective.

2.1 Federated Learning for Recommendation

Federated learning (FL) [8,33,34] utilizes distributed data to facilitate global training, thereby preserving participants' data from being learned. Specifically, federated learning operates on a distributed framework in which client-owned data is processed by local models instead of being uploaded to servers. In recommender systems, data leakage from servers compromises user privacy, as users' purchase histories are sensitive information [35].

Influenced by this, recent studies [8,12] consider using FL to preserve users' information and deliver personalized recommendation, leading to the proposal of FedRec [12,17,22,29]. In FedRec, the global model is developed by joint training with each user's local interactions. Each user acts as a client, maintaining a local recommendation model and uploading gradients to the server for integration. Specifically, FedCF [8] proposes the FedRec framework based on collaborative filtering (CF) [28,36], which trains user embeddings locally and updates item embeddings on the server. Following this, FedMF [12] introduces homomorphic encryption to address the issue of user privacy leakage due to continuous gradients. With the significant advancements of graph neural networks (GNNs) [9,28] in recommender systems, FedGNN [17] introduces a client-based graph collaborative training model, employing local differential privacy (LDP) [37] to ensure privacy preservation. To achieve personalized recommendation [38], PerFedRec [29] employs user clustering to divide clients into different groups and collaborates with a global model to fine-tune the local models. Furthermore, considering the sparsity of local data [19,21,39], PerFedRec++ [18] introduces a self-supervised pre-training enhanced FedRec framework, which constructs two global views generated by privacy-preserving mechanisms for contrastive learning to obtain initialized embeddings. Due to the heterogeneity of data in the real world [26,40], FedHGNN [22] further proposes a shared heterogeneous information framework that employs semantic protection and hierarchical attention mechanisms to achieve user personalization.

2.2 Contrastive Learning for Recommendation

As a paradigm for self-supervised learning, contrastive learning (CL) [27,41] focuses on learning invariant features by decreasing the distance between positive samples and increasing the distance between

negative samples. Specifically, contrastive learning utilizes data augmentation [39,40,42,43], such as structural perturbations and embedding disturbances, to construct multiple different views. It maximally captures consistency across perspectives through alignment operations.

Recently, some studies [38,39,44] have begun to merge contrastive learning with recommender systems, resulting in notable impacts. In recommender systems, data sparsity is a widely acknowledged issue. CL-based recommendation [45–47] effectively alleviate this limitation using self-supervised signals obtained from embedding alignment. For instance, SGL [39] designs an enhancement method based on structural perturbations (such as random edge dropping, node dropping) and first introduces the data augmentation-contrast paradigm. Following this foundation, NCL [46] observes the neighbor relationships between users and items, aligning outputs of these users and items across various GNN [17,28,48] layers to further extract supervised signals. Meanwhile, SimGCL [44] explores the effectiveness of complex structural enhancements, suggesting simple noise perturbations to bolster the contrastive paradigm. LightGCL [49] uses singular value decomposition (SVD) to construct lightweight contrastive views for recommendation. DirectAU [50] analyzes the alignment and uniformity properties of contrastive learning, using a loss function optimized for these objectives to effectively enhance recommendation performance.

2.3 Heterogeneous Graph Neural Network

Heterogeneous graphs (HGs) [24,26,40] more effectively model data from the real world by containing diverse types of edges and nodes. Recently, graph neural networks have demonstrated significant advantages in domains such as social networks [51] and recommender systems [52,53] due to their powerful capabilities of neighborhood aggregation and iterative propagation. Inspired by this, heterogeneous graph neural networks (HGNNs) [6,26,54] utilize meta-paths to capture diverse semantic information.

Specifically, HERec [6] integrates meta-path-based node embeddings with random walks and fusion functions. HAN [26] employs node-level and semantic-level attention mechanisms to capture the importance of nodes and meta-paths respectively. As self-supervised learning (SSL) [39,41,55] rapidly advances in various fields, SSL-based HGNNs [40,54] strive to explore unique self-supervised signals, achieving significant outcomes. For example, HeCo [40] first proposes a contrast between network schema and meta-path, and designs a HG-based sampling method to enhance this process. HGMAE [54] further introduces generative methods (such as masked autoencoders [56]), and designs dynamic masking mechanisms and position-aware encoding to enhance the model's semantic capturing ability. While these HGNNs are highly effective, they are designed for centralized data storage systems and are not appropriate for federated environments requiring privacy preserving.

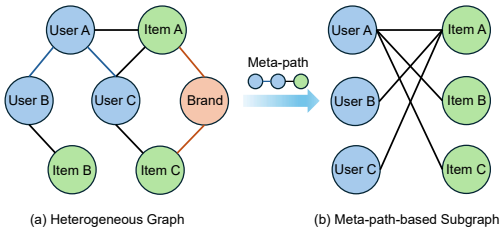


Fig. 2 A toy example of heterogeneous graph for recommendation. After the meta-path transformation, each type of user-item meta-path structure can be regarded as a subgraph based on specific semantics. This allows GNNs [28] to encode information from varying semantics. Best view in color.

3 PRELIMINARY

In this section, we introduce the definitions and concepts required in FedHGCL. The focus of the introduction is on the connection between clients and servers in HGs and privacy protection.

3.1 Heterogeneous Graph

In this section, we formally define some significant concepts related to heterogeneous graph as follows:

Definition 1 (Heterogeneous Graph (HG [26])): A HG is defined by the graph structure $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A}, \mathcal{R}, \phi, \varphi)$. In this structure, $\mathcal{V}$ and $\mathcal{E}$ represent the collections of nodes and edges, respectively. The type of each node v and edge e is determined by type mapping functions: $\phi : \mathcal{V} \rightarrow \mathcal{A}$ for nodes and $\varphi : \mathcal{E} \rightarrow \mathcal{R}$ for edges. Here, $\mathcal{A}$ symbolizes the various types of nodes, and $\mathcal{R}$ indicates the different types of edges. The sum of distinct node and edge types in the graph exceeds two, i.e., $|\mathcal{A}| + |\mathcal{R}| > 2$.

Definition 2 (Meta-path [40]): In a $\mathcal{G}$, a meta-path ρ is represented as $\mathcal{A}1 \xrightarrow{\mathcal{R}1} \mathcal{A}2 \xrightarrow{\mathcal{R}2} \dots \xrightarrow{\mathcal{R}l} \mathcal{A}l+1$, describing a composite connection between $\mathcal{A}1$ and $\mathcal{A}l+1$. As shown in Figure 2, connections are established between users (or items) in the figure on

Notation	Definition
$\mathbf{A}_i^{\rho k}$	The k-th meta-path for the i-th user.
$\mathbf{A}_i$	A set of updatable user i interaction subgraphs.
$\mathbf{E}_u; \mathbf{E}_i$	Initialized user and item node embeddings in server.
$\mathcal{N}_{u,i}$	The user IDs included in the client of user i.
$\mathbf{E}_{u,i}$	The required user embeddings in the client of user i.
$\mathbf{E}_{c_i}^k; \mathbf{E}_{c_i}$	The embedding obtained from the k-th subgraph; average pooling of them.
$\mathbf{H}_{c_i}^k; \mathbf{S}_{c_i}^k$	The IDs-based embeddings for users and items of clientc_i.
$\mathbf{H}_i^p; \mathbf{H}_i^q$	The target node of clientc_i; the positive sample of target node.
$\mathbf{e}_{c_i,m}; \mathbf{e}_{c_i,n}$	The local user embedding; the global item embedding.
$\hat{y}_{m,n}$	The client's prediction for user m on item n.
$\mathcal{B}$	The sampled batch for client c_i.
$\lambda_{bpr}; \lambda_{cl}; \lambda_\Theta$	The hyper-parameters controlling the losses for client c_i.
$\mathcal{L}_{bpr}; \mathcal{L}_{cl}; \|\Theta\|_2^2$	The BPR loss; the contrastive loss; the regularization loss.
Edges	The array of edges number in every client.
$C_{\text{top-}N}$	The top-N clients for gradient updates and data augmentation each epoch.
$\mathbf{g}_i^{(c_i)}; \mathbf{g}_u^{(c_i)}; \mathbf{g}_m^{(c_i)}$	The item embedding gradients; the user embedding gradients; the model gradients.
$\tilde{\mathbf{g}}^{(c_i)}; \bar{\mathbf{g}}$	The gradient uploaded by each client; the gradient to be updated by the server.
$\delta; \lambda; \eta$	The parameters for local differential privacy.
Θ_{global}^t	The Parameters to be updated by the server.
$\mathbf{s}_{user}^{m,n}; \mathbf{s}_{inter}^{m,n}$	The cosine similarity of user-user and user-item.

Table 1 Notations and Definitions in Our Proposed Framework.

the left according to the given meta-paths. For instance, user A and item B connect through the path "UserA-UserB-ItemB" to enrich the data. Different meta-paths usually reveal diverse interdependent information. The path "UUI" indicates that two socially connected users tend to like similar items.

Definition 3 (Meta-path-based Subgraph [57]): Based on the specific meta-path ρ, with user as the starting node type and item as the ending node type, users and items connect to form the bipartite graph that is stored in a matrix $\mathbf{R}^\rho \in \mathbb{R}^{M \times N}$, where M and N are the numbers of users and items, respectively. As depicted in Figure 2 (b), based on the aforementioned meta-path "UUI", a user-item interaction subgraph $\boldsymbol{G}^\rho = (\boldsymbol{V}^\rho, \boldsymbol{E}^\rho)$ is formed, with $\boldsymbol{V}^\rho$ representing all nodes and $\boldsymbol{E}^\rho$ comprising the connections of new user-item interactions.

3.2 Privacy Definition

In HGs, real historical interactions are stored in the adjacency matrix $\mathbf{R}^{M \times N}$, where M and N denote the number of users and items, respectively. Regarding heterogeneous information, the sparse interaction data is enriched using the meta-path-based subgraphs mentioned above. However, this data cannot be directly used; each user's interactions and personal information are locally stored, defined as follows:

Definition 1 (Client [19]): Each user u typically represents a client c, encompassing all interactions of a specific user within the HG. In the client c, interactions between users and items are defined as $\mathbf{R}_{n\cdot}$, and additional information as $\mathbf{C}_{n\cdot}$.

Definition 2 (Server [22]): The server coordinates training among all clients. It does not request client information about users' $\mathbf{R}_{n\cdot}$ and $\mathbf{C}_{n\cdot}$ but retrieves gradients or other necessary information for model updates after LDP is applied.

Definition 3 (Client Heterogeneity) [58]: Client heterogeneity refers to the system-level and data-level differences among clients in federated recommendation settings, including variations in local data volumes, computational and communication capabilities, and privacy perturbation strengths. This type of heterogeneity primarily affects the stability and efficiency of federated optimization and parameter aggregation.

Definition 4 (User Heterogeneity) [59]: User heterogeneity denotes the intrinsic diversity among users in terms of interest preferences, behavioral patterns, interaction frequencies, and historical behavior distributions. It reflects the semantic and behavioral non-uniformity of the recommendation task and constitutes a fundamental source of data sparsity and non-IID distributions across clients.

In this article, we assume that both clients and servers are honest but curious [12]. Overall, we cannot tamper with the training since the clients provide accurate information, and the real $\mathbf{R}_{n\cdot}$ and $\mathbf{C}_{n\cdot}$ of the clients are not uploaded to the server. Moreover, all server-side similarity computation and data augmentation in FedHGCL are performed solely on LDP-perturbed embeddings uploaded by clients, rather than raw interaction data. By the post-processing property of local differential privacy, these operations do not weaken the privacy guarantee, and the server cannot infer users' true preference patterns or interaction behaviors. Before introducing the proposed model, we summarize the symbols used in Table 1 in this paper. We present the symbols in the order of appearance for Section 4, 'The client-side' (Section 4.1), and 'The Server-side' Section (4.2) separated by lines.

4 METHODOLOGY

Figure 3 shows the overall framework of the proposed FedHGCL. Following the order in each training epoch, we first introduce the client's local recommendation module. Next, the server aggregates gradients from these clients and utilizes Local Differential Privacy (LDP) for privacy protection. Finally, the model uses privacy-preserving similarity calculations to provide feedback to each client for data augmentation to mitigate heterogeneity problem.

4.1 Client-side Heterogeneous Graph Contrastive Learning

Our proposed local recommendation module includes five key components: multi-view construction, embedding layer, graph convolution layer, embedding contrast, and ranking prediction.

4.1.1 Multi-view Construction

Following the definitions given in Section 3.1, we present the heterogeneous graph (HG) stored in the first client as $\mathcal{G}_1$, with its set of user-item interaction graphs based on different meta-paths denoted as $\left\{\mathbf{G}_1^{\rho 1}, \mathbf{G}_1^{\rho 2}, \cdots, \mathbf{G}_1^{\rho k}\right\}$, where k is the number of meta-paths we select. Next, these interaction subgraphs are stored in the corresponding matrix set $\left\{\mathbf{R}_1^{\rho 1}, \mathbf{R}_1^{\rho 2}, \cdots, \mathbf{R}_1^{\rho k}, \mathbf{R} \in \mathbb{R}^{M \times N}\right\}$ as adjacency matrices, with the specific formula as follows:

$$\mathbf{A}_M^{\rho k} = \begin{pmatrix} 0 & \mathbf{R}_M^{\rho k} \\ (\mathbf{R}_M^{\rho k})^T & 0 \end{pmatrix}, \tag{1}$$

where $\mathbf{A}_M^{\rho k}$ denotes the k-th meta-path for the M-th user. All users have the same number of subgraphs, constructed based on varying semantics. The linked users and items typically represent a specific relationship, for example, in Figure 3, $\mathbf{R}_1^{UIUI}$ for User 1 indicates that users who have purchased the same item have the same item preference. We describe the contrastive views constructed by all clients as follows:

$$\begin{cases} \mathbf{A}_1 = \left(\mathbf{A}_1^{\rho 1}, \mathbf{A}_1^{\rho 2}, \ldots, \mathbf{A}_1^{\rho k}\right), \\ \quad \cdots \\ \mathbf{A}_M = \left(\mathbf{A}_M^{\rho 1}, \mathbf{A}_M^{\rho 2}, \ldots, \mathbf{A}_M^{\rho k}\right), \end{cases} \tag{2}$$

We define $\mathbf{A}_i$ as a set of updatable user i interaction subgraphs, where $\mathbf{A}_i^{\rho} \in \mathbb{R}^{(M+N)\times(M+N)}$. Finally, $A = \{\mathbf{A}_1, \mathbf{A}_2, \cdots, \mathbf{A}_M\}$ refers to our client-side set of users.

4.1.2 Embedding Layer

Following prior work [19,28], we initialize user and item node embeddings, denoted as $\mathbf{E}_u \in \mathbb{R}^{M \times d}$ and $\mathbf{E}_i \in \mathbb{R}^{N \times d}$, where d represents the embedding dimension. Additionally, considering privacy, each user independently stores their embedding in client c_i, and the item embeddings $\mathbf{E}_i$ information is shared and iteratively updated among users through the server. After obtaining their embedding vectors $\mathbf{e}_u$ and item embeddings, clients need to train their local GNN models using the user-item interaction matrix. Given that this interaction data is sensitive, it is inappropriate

to share it openly between servers or among users. For this reason, our study adopts the strategy from document [17,18], where users upload their embedding vectors and a uniformly encrypted ID to the server. Next, the server provides users with encrypted project IDs and embedding information. Finally, the client c_i requests from the server the embedding tables for all required users and items. The formula for obtaining user embeddings is as follows:

$$\mathbf{E}_{u,i} = \begin{cases} \mathbf{E}_{u,i,j} & \text{if } i \in \mathcal{N}_{u,i} \\ \mathbf{0} & \text{otherwise} \end{cases} \tag{3}$$

where $\mathbf{E}_{u,i,j}$ is the embedding value at the j-th dimension of user i in the original embedding matrix $\mathbf{E}_u$. Additionally, $\mathcal{N}_{u,i}$ includes all user IDs included in the client of user i, where $\mathbf{0}$ denotes a zero vector with a dimension d. This method complies with the data minimization principle (that is, processing only necessary data), a requirement of numerous privacy laws, such as the GDPR.

4.1.3 Graph Convolution Layer

In this module, the input for each client c_i is the corresponding matrix $\mathbf{E}_i$ and embedding $\mathbf{E}_{u,i}$. The encoder here can be any GNN, such as GAT [26], GCN [36]. Following classical works [18,28], we encode each client's information through a lightweight graph convolution, whose simplified structure allows the model to focus more on capturing higher-order connections between users and items. The specific encoding process for each client is as follows:

$$\begin{cases} \mathbf{E}_{c_i}^1 = \mathbf{A}_i^{\rho 1} \cdot f_{concat}\left(\mathbf{E}_{u,i}, \mathbf{E}_i\right), \\ \quad \cdots \\ \mathbf{E}_{c_i}^k = \mathbf{A}_i^{\rho k} \cdot f_{concat}\left(\mathbf{E}_{u,i}, \mathbf{E}_i\right), \end{cases} \tag{4}$$

where $\mathbf{E}_{c_i}^k$ denotes the embedding obtained from the k-th subgraph encoding of client c_i, and the function $f_{concat}(,)$ is the direct concatenation of two vectors. $\mathbf{A}_i^{\rho k}$ is the k-th semantic subgraph based on user i, mentioned in Section 4.1.1. Considering that multiple GCN layers might introduce noise, we set all layers to 1. Next, we apply average pooling to the embeddings learned from each subgraph as the input for the final prediction:

$$\mathbf{E}_{c_i} = \text{Pooling}(\{\mathbf{E}_{c_i}^1, \cdots, \mathbf{E}_{c_i}^k\}) = \frac{1}{k}\sum_{n=1}^{k} \mathbf{E}_{c_i}^n, \tag{5}$$

where $\mathbf{E}_{c_i}$ is obtained through an average pooling operation. This operation aids in smoothing and integrating diverse semantic information, potentially effective in mitigating accidental semantic noise impacts, while preserving the information learned by the GCN from various subgraphs.

4.1.4 Contrastive Learning

We retrieve the embeddings contained in client c_i from $\{\mathbf{E}_{c_i}^1, \cdots, \mathbf{E}_{c_i}^k\}$ using the IDs of users and items, resulting in **user embedding**: $\{\mathbf{H}_{c_i}^1, \cdots, \mathbf{H}_{c_i}^k\}$ and **item embedding**: $\{\mathbf{S}_{c_i}^1, \cdots, \mathbf{S}_{c_i}^k\}$. We select two embeddings from the generated set that do not belong to the same subgraphs p and q, where $p \neq q$. For each $\mathbf{H}_i^p$, select $\mathbf{H}_i^q$ as the positive sample, and $\mathbf{H}_j^p$ selected from users and items not included in client c_i as the negative sample. We then use the InfoNCE [27] loss formula to compute the loss for the entire batch:

$$\begin{aligned} \mathcal{L}_{c_i}^{user} &= -\sum_{i\in\mathcal{B}} \log\left(\frac{\exp(\text{sim}(\mathbf{h}_i^p, \mathbf{h}_i^q)/\tau^{'})}{\sum_{j\in\mathcal{B}}\exp(\text{sim}(\mathbf{h}_i^p, \mathbf{h}_j^q)/\tau^{'})}\right), \\ \mathcal{L}_{c_i}^{item} &= -\sum_{i\in\mathcal{B}} \log\left(\frac{\exp(\text{sim}(\mathbf{s}_i^p, \mathbf{s}_i^q)/\tau^{''})}{\sum_{j\in\mathcal{B}}\exp(\text{sim}(\mathbf{s}_i^p, \mathbf{s}_j^q)/\tau^{''})}\right), \end{aligned} \tag{6}$$

where $\text{sim}(x, y)$ is the similarity function between x and y, and i, j are users/items in a sampled batch $\mathcal{B}$. The $\tau^{'}$ and $\tau^{''}$ denotes the temperature coefficient used to control the smoothness of the softmax function. This contrastive method reduces the embedding distances across multiple semantics while simultaneously minimizing similarity with negative samples [39,44]. By employing contrastive learning-based multiple views, we capture additional supervisory signals, helping alleviate performance degradation caused by data sparsity on the clients. Finally, the CL-loss for each client c_i is the sum of the losses of users and items:

$$\mathcal{L}_{cl}^{c_i} = \mathcal{L}_{c_i}^{user} + \mathcal{L}_{c_i}^{item}, \tag{7}$$

This multi-view alignment allows for a more comprehensive utilization of available data, thereby enhancing the model's robustness when dealing with sparse data. Specifically, even if some users or items have sparse interaction data, the model can learn their features by contrasting the similarities and differences between various semantics.

4.1.5 Optimization Loss

We use the dot product of the local user embedding $\mathbf{e}_{c_i,m}$ and the global item embedding $\mathbf{e}_{c_i,n}$ (from Section 4.1.3) to predict local historical interactions. Specifically, the model's prediction for user m on item n is denoted as:

$$\hat{y}_{m,n} = \mathbf{e}_{c_i,m} \cdot \mathbf{e}_{c_i,n}, \tag{8}$$

Through the above process, we employ the typical

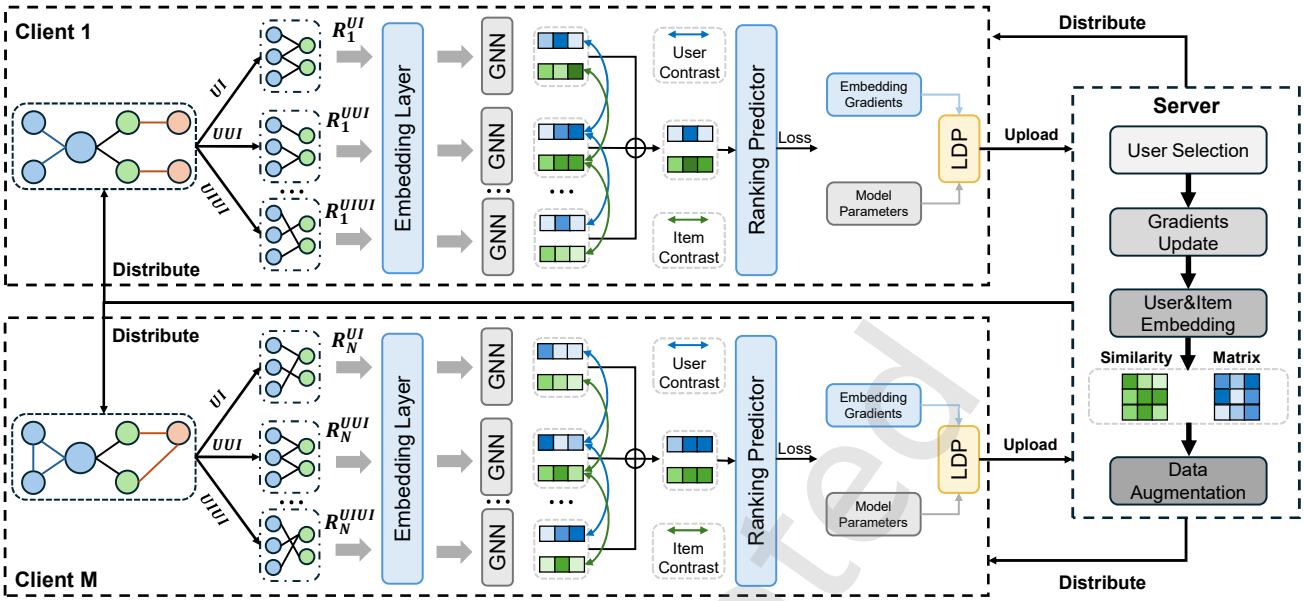


Fig. 3 The complete framework of the proposed FedHGCL. The model contains two important modules: (a) Client: we employ meta-paths to construct the heterogeneous graph into multiple interactive perspectives, the contrast between perspectives helps clients automatically mine supervision signals. (b) Server: we aggregate the model gradients with LDP through a user selection strategy and provide a data augmentation method as feedback to the clients. Best view in color.

Bayesian Personalized Ranking (BPR) [60] loss function to optimize parameters of client c_i:

$$\mathcal{L}_{bpr}^{c_i} = -\frac{1}{|\mathcal{B}|} \sum_{(m,n,k)\in B} \log \sigma\left(\hat{y}_{m,n} - \hat{y}_{m,k}\right), \quad (9)$$

where $\mathcal{B} = \{(m,n,k) \mid A_{m,n} = 1, A_{m,k} = 0\}$ is the training data. This loss function enforces that the predicted scores for observed interactions are higher than those for unobserved interactions. Finally, the complete optimization objective of client c_i is as the follow:

$$\mathcal{L}_{ci} = \mathcal{L}_{bpr}^{c_i} + \lambda_{cl} \cdot \mathcal{L}_{cl}^{c_i} + \lambda_{\Theta} \cdot \|\Theta\|_2^2, \quad (10)$$

where λ_{cl} and λ_{Θ} are hyper-parameters we set, and $\|\Theta\|_2^2$ are trainable model parameters and L_2 regularization.

4.2 Server-side Aggregation and Data Augmentation (DA)

In this section, we describe the structure of the FedHGCL server according to the order in Figure 3: user selection, gradient updating, and data augmentation.

4.2.1 User Selection based on Client Data

To augment data for clients with sparser data, we propose a data-based user selection strategy. First, for each client c_i, as defined in Section 3.1, let $\mathcal{G}_{c_i}$ be the corresponding heterogeneous graph. It is known that $\mathcal{E}_{c_i}$ is the set of edges in the graph $\mathcal{G}_{c_i}$, thus the total number of edges for client c_i is $|\mathcal{E}_{c_i}|$. Therefore, we construct an array Edges, where each element is a tuple containing the client's identifier and the number of edges in that client's graph:

$$\text{Edges} = [(c_1, |\mathcal{E}(\mathcal{G}_{c_1})|), (c_2, |\mathcal{E}(\mathcal{G}_{c_2})|), \cdots, (c_M, |\mathcal{E}(\mathcal{G}_{c_M})|)], \quad (11)$$

Next, we sort the array Edges in ascending order based on the number of edges in each client's graph. Finally, we select the top-N clients for gradient updates and data augmentation. The specific process is as follows:

$$C_{\text{top-}N} = (\text{sort}\,(\text{Edges}, \text{key} = \text{lambda}\ x : x[1]))\,[:N], \quad (12)$$

where $\text{sort}(\cdot,\cdot)$ is a sorting operation, where x represents each element in the Edges array (i.e., each tuple), and $x[1]$ refers to the second element of the tuple, that is, $|\mathcal{E}(\mathcal{G}_{c_i})|$, meaning it sorts by the number of edges. The $[:N]$ is a slicing operation used to obtain the first N elements from the list.

4.2.2 Aggregation based on Local Differential Privacy

Direct transmission of embedding gradients and model gradients can lead to privacy leaks, thus it is necessary to safeguard them [12]. Given that each client maintains a full embedding table for items, using homomorphic encryption for gradient protection is impractical due to significant storage and communication costs; therefore, we adopt the

LDP method proposed by [17] for safeguarding local gradients.

Specifically, we instantiate the item embedding gradient, user embedding gradient, and model gradient from client c_i as $\mathbf{g}_t^{(c_i)}$, $\mathbf{g}_u^{(c_i)}$, and $\mathbf{g}_m^{(c_i)}$, respectively. Once merged, the gradient is defined as $\mathbf{g}^{(c_i)} = \{\mathbf{g}_t^{(c_i)}, \mathbf{g}_u^{(c_i)}, \mathbf{g}_m^{(c_i)}\} = \frac{\partial \mathcal{L}_{c_i}}{\partial \Theta}$, with Θ denoting all trainable parameters. Thus, the formula for LDP is:

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

where $\text{clip}(x, \delta)$ means to restrict the gradient x within a certain range. Specifically, if the absolute value of an element of the gradient exceeds δ, it is clipped to δ. This measure is taken to avoid excessively large gradients that could destabilize the training process. $\text{Laplacian}(0, \lambda)$ indicates noise that is sampled from a Laplace distribution. The Laplace distribution is centered at zero, with λ representing the scale parameter's proportionality factor for Laplacian noise. In each round, the server connects with a batch of clients ($C_{\text{top-}N}$) and first sends the current model gradient $\tilde{\mathbf{g}}$ to these clients:

$$\bar{\mathbf{g}} = \sum_{i \in C_{\text{top-}N}} \frac{D_i}{\sum_{j \in C_{\text{top-}N}} D_j} \tilde{\mathbf{g}}, \tag{14}$$

where D_i is the local data amount in client c_i, and $C_{\text{top-}N}$ is the set of selected clients. The server updates the parameter Θ_{global}^t using gradient descent to:

$$\Theta_{global}^t = \Theta_{global}^{t-1} - \eta \cdot \bar{\mathbf{g}}, \tag{15}$$

where η is the learning rate. This training process is repeated multiple times until convergence.

4.2.3 Data Augmentation based on Similarity

Our data augmentation process is divided into user-based and interaction-based. First, we calculate similarities using privacy-protected user embeddings $\tilde{\mathbf{E}}_u = \{\tilde{\mathbf{e}}_u^1, \tilde{\mathbf{e}}_u^2, \cdots, \tilde{\mathbf{e}}_u^M\}$ and public item embeddings $\mathbf{E}_i = \{\mathbf{e}_i^1, \mathbf{e}_i^2, \cdots, \mathbf{e}_i^N\}$ on the server:

$$\begin{aligned} \mathbf{s}_{user}^{m,n} &= \mathbf{s}(\tilde{\mathbf{e}}_u^m, \tilde{\mathbf{e}}_u^n) = \frac{\tilde{\mathbf{e}}_u^m \cdot \tilde{\mathbf{e}}_u^n}{\sqrt{(\tilde{\mathbf{e}}_u^m)^2} \times \sqrt{(\tilde{\mathbf{e}}_u^n)^2}}, \\ \mathbf{s}_{inter}^{m,n} &= \mathbf{s}(\tilde{\mathbf{e}}_u^m, \mathbf{e}_i^n) = \frac{\tilde{\mathbf{e}}_u^m \cdot \mathbf{e}_i^n}{\sqrt{(\tilde{\mathbf{e}}_u^m)^2} \times \sqrt{(\mathbf{e}_i^n)^2}}. \end{aligned} \tag{16}$$

where m and n refer to the similarity between user m and user n in the first equation, and user m for item n in the second equation. After obtaining the similarity matrix, we select the top-K users and user-item interactions for each client c_i in the selected client set $C_{\text{top-}N}$ to incorporate into the subsequent training round. To ensure the diversity of the multi-view construction, we will pass heterogeneous information $\mathbf{C}_{n\cdot}$ (Section 3.2) to each client based on the index of the users and items. This information will be used to construct new multi-view contrasts for the next round of distributed training.

4.2.4 Algorithm

The pseudo-code of the FedHGCL algorithm is shown in Algorithm 1. Before each round of training, clients must construct multi-view subgraphs from the heterogeneous graphs distributed by the server for contrastive learning in the local recommendation module. Each client's update is detailed in the function "ClientUpdate()" below the algorithm, which describes the detailed process from the input local graph to the gradients. Within the algorithm, the loop from lines 7 to 14 runs on the server, during which the server dispatches parameters to the clients and gathers their gradients for updating.

4.2.5 Time complexity

From a computational perspective, the time complexity of FedHGCL can be analyzed per communication round from both the client and server sides. On the client side, each participating client mainly performs multi-view subgraph encoding, contrastive learning, and local ranking optimization, resulting in an aggregated complexity of

$$\mathcal{O}\left(\sum_{i \in C} \left(k|E_i|d + |B|^2 d + |B|d + |\Theta|\right)\right) \tag{17}$$

where $|E_i|$ denotes the number of edges in the local subgraph and $|B|$ is the mini-batch size. On the server side, the computation is dominated by client selection, gradient aggregation, and similarity-based data augmentation, whose overall cost scales as $\mathcal{O}(M \log M + n(M+N)d + n|\Theta|)$, avoiding the quadratic overhead of constructing a full similarity matrix. Combining both sides, the overall time complexity of one communication round is

$$\mathcal{O}\left(kd \sum_{i \in C} |E_i| + n|B|^2 d + M \log M + n(M+N)d + n|\Theta|\right) \tag{18}$$

and the total training complexity scales linearly with the number of communication rounds T.

5 EXPERIMENT

In this section, to better understand and optimize our FedHGCL framework, we pose the following research questions:

Algorithm 1 FedHGCL (Federated Heterogeneous Graph Contrastive Learning)

Input: Embedding Size, learning rate: d, η
  Total number of clients, items: M, N
  LDP parameter: δ, λ, η
  Clients local graph: $\{\mathbf{A}_i\}_{i=1}^{M}$
Output: Model parameters and embeddings Θ; Local client embeddings $\{\mathbf{E}_{c_i}\}_{c_i=1}^{M}$

Initialize Θ**;**
while not converge **do**
  user selection $C_{\text{top-}N} \leftarrow$; Equation 11 and 12 ▷ client selection
  For $c_i \in C_{\text{top-}N}$ **do**
    $\mathbf{g}_{c_i} \leftarrow \text{ClientUpdate}(c_i, \Theta)$; ▷ collect gradients from clients
    $\bar{\mathbf{g}}_{c_i} \leftarrow$ Equation 14; ▷ average gradients from clients
    $\Theta_{global}^{t} \leftarrow \Theta_{global}^{t-1} - \eta \cdot \bar{\mathbf{g}}$; ▷ update parameters
end while

Function ClientUpdate(c_i, Θ)
  downloading Θ and augmented data $\mathbf{C}_{n\cdot}$ from server;
  constructing multi-view subgraphs $\leftarrow$ Equation 1, 2 and 3;
  $\mathbf{E}_{c_i} \leftarrow$ Equation 4 and 5; ▷ local user GNN
  $\mathcal{L}_{cl}^{c_i} \leftarrow$ Equation 6 and 7; ▷ calculate contrastive loss
  $\mathcal{L}_{bpr}^{c_i} \leftarrow$ Equation 8 and 9; ▷ calculate bpr loss
  $\mathcal{L}_{c_i} \leftarrow$ Equation 10; ▷ calculate local recommendation loss
  $\mathbf{g}^{(c_i)} = \frac{\partial \mathcal{L}_{c_i}}{\partial \Theta}$; ▷ calculate the gradients
  $\tilde{\mathbf{g}}^{(c_i)} \leftarrow$ Equation 13; ▷ LDP for gradients
  return $\tilde{\mathbf{g}}^{(c_i)}$ ▷ return gradients

- **RQ1:** How does FedHGCL compare to the current state-of-the-art (SOTA) models in terms of federated recommendation performance?
- **RQ2:** What are the reasons for FedHGCL's superior performance, and how does it differ from existing models?
- **RQ3:** Are the key components in our FedHGCL delivering the expected performance gains?
- **RQ4:** How do different hyperparameter settings affect FedHGCL?

5.1 Experiment Setup

The specific experimental setup is described in terms of datasets, baseline methods, and implementation details.

5.1.1 Datasets

We select a publicly available HG-based recommendation dataset, following setup [44], and split it into training, validation, and test sets in an 8:1:1 ratio [61,62]. All training data are stored on local clients as users' private data . The global graph information is obtained using reconstruction methods and does not contain real user information.

- **Last.FM**[‡]. The music recommendation dataset contains three types of nodes: user, artist, and tag, along with four edge relationships. We predict whether there is an interaction between a user and an artist.
- **Yelp** [6]. The business recommendation dataset contains five types of nodes, including user, business, and city, along with five edge relationships. We predict whether there is an interaction between a user and an business.
- **Douban Movie**[§]. The movie recommendation dataset contains six types of nodes, including user, movie, and actor, as well as six edge relationships. We predict the interactions between a user and a movie.

5.1.2 Baselines

We select the most representative baseline models for comparison, categorizing them into methods based on centralized learning and federated learning. Specifically, **centralized Learning Models:** MF [11], HAN [26], LightGCN [28], SGL [39] and HGCL [38]. **Federated Learning Models:** FedMF [12], FedGNN [17], FeSoG [19], PerFedRec [29], PerFedRec++ [18] and FedHGNN [22].

General Recommended Models.

- MF [11] learns implicit feature vectors between users and items to optimize differences in pairwise preferences.
- HAN [26] employs node-level and semantic-level attention mechanisms to capture the importance of nodes and meta-paths effectively.
- LightGCN [28] simplifies GCN architecture by eliminating feature transformations and nonlinear activations to streamline recommendation.
- SGL [39] generates contrasting views through edge dropout, enhancing recommendation quality via contrastive learning.

[‡]https://www.last.fm/
[§]https://m.douban.com/

# Dataset	# Nodes	# Private/ Shared Links	# Meta-paths
Last.FM	User (U): 1,892	U-A: 92,834	U-U-A
	Artist (A): 17,632	U-U: 25,434	U-A-T-A
	Tag (T): 11,945	A-T: 184,941	U-A-U-A
Yelp	User (U): 16,239	U-B: 198,397	U-U-B
	Business (B): 14,284	U-U: 158,590	U-B-C-B
	Category (C): 511	B-C: 40,009	U-B-U-B
Douban Movie	User (U): 13,367	U-M: 1,068,278	U-U-M
	Movie (M): 12,677	U-U: 4,085	U-M-A-M
	Actor (A): 6,311	M-A: 33,587	U-M-D-M
	Director (D): 2,449	M-D: 11,276	U-M-U-M

Table 2 Dataset statistics.

- HGCL [38] integrates heterogeneous relational semantics by leveraging contrastive self-supervised learning to enhance recommender systems.

Federated Learning Models.

- FedMF [12] employs a secure aggregation protocol, allowing multiple parties to collaboratively train mid-frequency models on their private data without sharing it.
- FedGNN [17], a GNN-based FedRec framework, obtains higher-order user neighbors via a trusted third-party server and uses pseudo-interaction sampling to protect privacy.
- FeSoG [19] is a federated social recommender model that employs relational attention and aggregation to manage heterogeneity, using local data to infer user embeddings for maintaining personalization.
- PerFedRec [29] performs user clustering on the server and aggregates parameters within each cluster to facilitate personalization.
- PerFedRec++ [18] utilizes privacy-preserving mechanisms to generate two augmented graph views for self-supervised pre-training, enhancing the consistency of representation learning and thus boosting the performance of the joint model.
- FedHGNN [22] proposes a novel semantic-preserving perturbation approach for releasing user interaction information to provide recommendation.

5.1.3 Implementations

For a fair comparison, our sampling methods and dataset formats follow classical works [18,44]. Every model, including the baseline ones, is retrained to convergence with embeddings initialized via Xavier [63] and optimized by Adam [64]. For some classic HGNN-based models, such as HAN [26] and HERec [6], we replace the cross-entropy loss [40] with BPR loss [60]. On each dataset, we tune the hyper-parameters for the baseline to ensure optimal performance. For general settings, GCN-based models [17,29] maintain layer numbers, with a learning rate of 0.001, an embedding size of 64, and a batch size of 2048. We analyze parameters unique to FedHGCL in RQ4. To clarify the implementation details of meta-path selection, we emphasize that all meta-paths used in our framework are manually specified rather than automatically discovered. For each dataset, we explicitly list the adopted meta-paths in Table R2 and further explain their semantic meanings.

5.2 Performance Comparison (RQ1)

In this section, Tables 3 and 4 report the recommendation performance of all baseline models on three public datasets. The experimental results are categorized into two types: centralized and federated. Based on the outcomes of the evaluation metrics, we have summarized potential explanations for these results. We summarize the conclusions are as follows:

- Our proposed method significantly outperforms the most advanced federated recommendation methods across all datasets. Compared to the baseline (eg.,FedHGNN [22], PerFedRec++ [18]), FedHGCL achieves an relative improvement of

9.61% and 9.16% in top-10 recall and NDCG on Yelp dataset, respectively. This improvement demonstrates the necessity of constructing multiple views of users and items for comparison using heterogeneous information.

- GNN-based methods (e.g., LightGCN [28]) can directly model structural information and learn high-order collaborative signals, making them generally superior to non-GNN methods (e.g., MF [11]). Moreover, FedGNN significantly outperforms FedMF in federated learning baselines, further demonstrating the superiority of GNN-based models over MF-based methods.

- CL-based methods, such as SGL [39], are clearly superior to other GNN-based methods, a trend already recognized in traditional recommendation. We attribute this performance enhancement to the contrast between multiple views, which help to maximize the consistency of representations across different views. Additionally, PerFedRec++'s clear superiority over FedGNN also demonstrates the effectiveness of contrastive learning in alleviating data sparsity.

- Centralized methods achieve the best results in nearly all cases. However, compared to centralized learning, federated learning shows a decline in performance. For privacy concerns, federated learning frameworks lack access to local data, limiting their ability to model global structures. Due to the heterogeneity of the clients, the imbalance in gradient updates makes it difficult for the server to receive high-quality gradients. We employ a sampling strategy to mitigate this imbalance and combine it with a data augmentation strategy to further alleviate the data island effect. Our FedHGCL outperforms all federated methods and surpasses most traditional centralized recommendation methods, proving the advantage of this strategy.

5.3 Explainability and Visualization (RQ2)

In this section, we present an example of data augmentation and explain its role in contrastive learning based on heterogeneous graphs.

5.3.1 Case Study

Figure 4 depicts the heterogeneous graphs constructed by user 13330 before and after data augmentation, where only part of the updated result is shown due to its excessive density. First, on the left side of the figure, the initial graph is very sparse, containing only a few key relationships (such as user-user and user-item). On the right side, after data augmentation, part of the graph shows dense key interactions. The above observation leads to a conclusion: a single round of enhancement often brings a large amount of interaction and heterogeneous information. However, are such interactions advantageous? We elucidate this with several findings below:

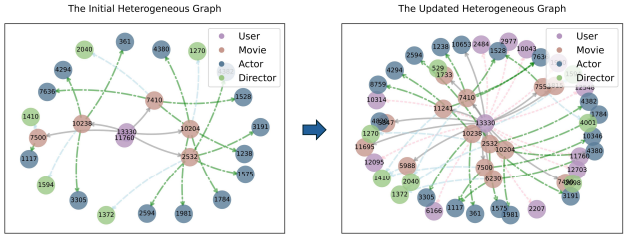


Fig. 4 A case study of data augmentation for user 13330. Best view in color.

- We can see that the graph on the right not only introduces user-item interactions but also connects some items through the meta-paths 'movie-actor-movie' and 'movie-director-movie'. This is explainable in recommendations, as users may prefer movies featuring the same actors or directed by the same directors.

- Additionally, using meta-paths to create multiple views on such graphs expands the receptive field of view for each view. During subsequent contrast learning process, these expanded perspectives are able to capture more information, thus enhancing the model's performance and accuracy.

5.3.2 Theoretical Analyses

Following the perspective proposed by [39], we further verify that more viewpoints are key to improving the performance of the local recommendation module. First, we investigated the self-supervised loss in Equation 6 and identified a reason: it inherently has the capability for hard negative mining, which contributes a significant amount of meaningful gradients to optimization and guides node representation learning. Formally, for node $u \in \mathcal{B}$, the gradient of the self-supervised loss with respect to the representation $\mathbf{h}_i^p$ is

Method	Last.FM		Yelp		Douban Movie	
	Recall	NDCG	Recall	NDCG	Recall	NDCG
MF [11]	0.1732	0.2124	0.0475	0.0392	0.1123	0.1791
HERec [6]	0.1714	0.2103	0.0470	0.0388	0.1112	0.1774
HAN [26]	0.1758	0.2158	0.0482	0.0398	0.1140	0.1819
LightGCN [28]	0.1876	0.2366	0.0594	0.0481	0.1149	0.2017
SGL [39]	0.1933	0.2414	0.0632	0.0519	0.1212	0.2035
HGCL [38]	0.1917	0.2400	0.0638	0.0522	0.1190	0.2031
FedMF [12]	0.1693	0.2078	0.0463	0.0383	0.1098	0.1747
FedGNN [17]	0.1709	0.2104	0.0468	0.0387	0.1112	0.1766
PerFedRec [29]	0.1721	0.2118	0.0473	0.0390	0.1120	0.1784
FeSoG [19]	0.1744	0.2146	0.0479	0.0396	0.1134	0.1808
PerFedRec++ [18]	0.1765	0.2172	0.0484	0.0401	0.1146	0.1829
FedHGNN [22]	0.1784	0.2191	0.0489	0.0404	0.1157	0.1846
FedHGCL	**0.1838**	**0.2309**	**0.0536**	**0.0441**	**0.1169**	**0.1897**
Improvement	3.03%	5.39%	9.61%	9.16%	1.04%	2.76%
p-value	1.5e-5	3.9e-3	3.7e-5	2.6e-4	2.2e-3	2.0e-4

Table 3 Performance Comparison on Last.FM, Yelp and Douban Movie on Top-10 prediction. The best result is in bold.

as follows:

$$\frac{\partial \mathcal{L}_{c_i}^{user}}{\partial \mathbf{h}_i^p} = \frac{1}{\tau' \|\mathbf{h}_i^p\|} \left(c(u) + \sum_{v \in \mathcal{B} \setminus \{u\}} c(v) \right), \quad (19)$$

where $c(u)$ and $c(v)$ denote the contributions of the positive node u and the negative nodes v to the gradient. This suggests that hard negative nodes provide larger gradients for guiding optimization, thereby enhancing the discriminative power of users' and items' representations. More perspectives provide more hard negative nodes, which is crucial for reducing training time and improving recommendation performance [65]. Meanwhile, a single perspective may contain noise or incomplete information, while the fusion of multiple perspectives can mitigate the shortcomings of a single perspective and enhance model's robustness. Based on the above case study, we can demonstrate that the results of data augmentation significantly contribute to the multi-view contrasts.

5.4 Ablation Study (RQ3)

In this section, we verify the effectiveness of key components in FedHGCL and provide possible explanations for the results. Here are our interpretations of various model variants.

- FedHGCL$_{\text{w/o CL}}$: It removes the contrastive loss and relies solely on multi-view embedding fusion for the recommendation task.
- FedHGCL$_{\text{w/o HGCL}}$: It solely utilizes client-side user-item interactions for local recommendations, and correspondingly, data augmentation of heterogeneous information is also canceled.
- FedHGCL$_{\text{w/o US}}$: It does not perform user sampling but instead conducts gradient uploading and updating across all clients.
- FedHGCL$_{\text{w/o DA}}$: It removes the server's data augmentation module, meaning it does not distribute augmented heterogeneous information to clients.

Table 5 shows the experimental results of all variants on all datasets, and we have the following findings:

- Given the independence of modules, our experimental results involve training with just a single module removed. It can be observed that the HGCL module significantly contributes to improving recommendation performance, emphasizing the necessity of utilizing heterogeneous information from a contrastive learning perspective to alleviate data sparsity on the client-side. In contrast, the CL module plays a secondary role, used to capture similarities across multiple views to mine supervisory signals. However, this phenomenon further illustrates that the natural solution is to use HG to enrich sparse interactions.
- Subsequently, we further divided the client module

Method	Last.FM		Yelp		Douban Movie	
	Recall	NDCG	Recall	NDCG	Recall	NDCG
MF [11]	0.2437	0.2446	0.0758	0.0421	0.1736	0.1806
HERec [6]	0.2392	0.2507	0.0744	0.0450	0.1702	0.1784
HAN [26]	0.2478	0.2395	0.0776	0.0388	0.1767	0.1834
LightGCN [28]	0.2636	0.2638	0.0893	0.0565	0.1791	0.2011
SGL [39]	0.2724	0.2704	0.0971	0.0614	0.1890	0.2056
HGCL [38]	0.2708	0.2698	0.0973	0.0615	0.1859	0.2047
FedMF [12]	0.2421	0.2537	0.0753	0.0455	0.1723	0.1805
FedGNN [17]	0.2430	0.2493	0.0755	0.0438	0.1730	0.1806
FeSoG [19]	0.2459	0.2423	0.0767	0.0405	0.1752	0.1820
PerFedRec [29]	0.2443	0.2460	0.0761	0.0421	0.1763	0.1832
PerFedRec++ [18]	0.2544	0.2532	0.0823	0.0473	0.1804	0.1885
FedHGNN [22]	0.2565	0.2528	0.0837	0.0484	0.1780	0.1926
FedHGCL	**0.2602**	**0.2598**	**0.0870**	**0.0515**	**0.1813**	**0.1991**
Improvement	1.44%	2.61%	4.00%	6.40%	0.50%	3.37%
p-value	3.7e-4	8.3e-3	1.9e-4	5.8e-5	7.6e-3	4.1e-5

Table 4 Performance Comparison on Last.FM, Yelp and Douban Movie on Top-20 prediction. The best result is in bold.

Method	Last.FM		Yelp		Douban Movie	
	Recall	NDCG	Recall	NDCG	Recall	NDCG
w/o CL	0.2510	0.2517	0.0821	0.0480	0.1755	0.1923
w/o HGCL	0.2473	0.2480	0.0780	0.0446	0.1735	0.1895
w/o US	0.2579	0.2565	0.0857	0.0503	0.1789	0.1956
w/o DA	0.2558	0.2546	0.0840	0.0504	0.1784	0.1960
FedHGCL	**0.2602**	**0.2598**	**0.0870**	**0.0515**	**0.1813**	**0.1991**

Table 5 Performance of different variants of FedHGCL on three datasets on Top-20 prediction.

into user sampling (US) and DA to study their roles. It is noteworthy that DA typically exerts a more significant influence on performance. This demonstrates that similarity-based graph augmentation and distribution process effectively enhances local training on the client-side. Finally, US, as a key component for balancing data distribution and reducing communication costs, takes into account the impact of data volume on distribution, thus validating that this strategy effectively mitigates client heterogeneity issues.

5.5 Hyper-parameter Study (RQ4)

In this section, we will analyze the impact of hyper-parameters from several key aspects. This specifically includes user's batch size, data auamentation, embedding dimension, learning rate and local differential privacy parameters.

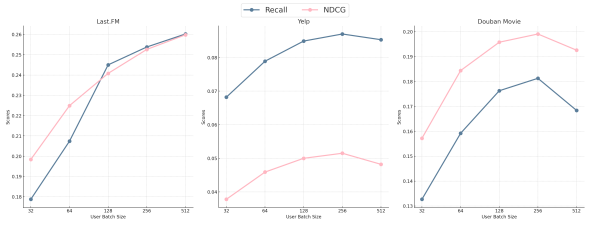


Fig. 5 Performance comparison of different user batch size on three datasets w.r.t. recall and NDCG. Best view in color.

5.5.1 User Selection Batch Size

In this section, we report the results of the top-20 recommendations on three datasets for the N clients mentioned in Section 4.2.1. First, it is noted that the model often performs better when larger batch sizes such as 256 or 512 are selected. Moreover, there is a significant gap between small and large sizes. We believe this phenomenon is due to the increased number of clients per training round, allowing the server to receive more gradients, thereby mitigating the limitations of gradient imbalance.

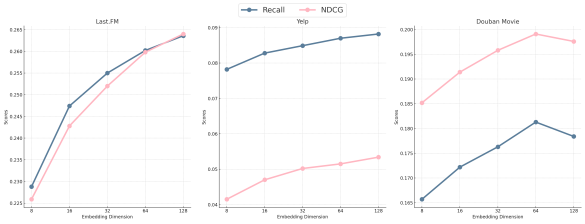


Fig. 6 Performance comparison of different embedding dimension on three datasets w.r.t. recall and NDCG.

In practical applications, increasing the number of participants in each training step allows the server to obtain a more accurate global information estimate, thereby enhancing performance. However, this approach increases computational costs and may prolong the time it takes for the model to converge. It is necessary to achieve a balance between training costs and recommendation performance.

5.5.2 Embedding Dimension

We explored the impact of embedding dimensions on federated recommendation performance, ranging from 16 to 256. In Figure 5, we observed that apart from the highest performance at dimension 64 for the Douban Movie dataset, the performance of other datasets consistently improved at dimension 128.

However, considering the decreasing rate of improvement and the increased training time and memory costs due to larger dimensions, we decided to fix the dimensions at 64 for all experiments. Moreover, due to the risk of overfitting in graph neural networks, the increase in dimensions is limited.

5.5.3 Data Auamentation

The parameters here correspond to the top-K interactions in Section 4.2.3 used for data augmentation distributed to clients. Figure 7 reports the results for K=10, 20, and 40 on each dataset. Notably, Last.FM and Douban Movie exhibited the same trend, achieving their best results at K=20. We attribute these results to the effect of sparsity. To my knowledge, Yelp is a very sparse dataset, thus requiring more data for augmentation.

However, the process of distributing data requires substantial computational and communication resources. Therefore, we only experimented with smaller K values in our model. Second, the pseudo-interactions utilized by FeSog and FedGNN are intended only for privacy protection and do not affect performance [18, 19]. Meanwhile, our Data Augmentation (DA) is crucial for enhancing performance, as mentioned in Sections 5.3 and 5.4. Overall, this experiment demonstrates the effectiveness of DA and its significant advantage over the pseudo-interactions used by other models.

5.5.4 Learning Rate and Communication Step

In this section, we explore the impact of the learning rate and communication step in the convergence process of the federated learning framework. As shown in Figure 8, we compared the convergence efficiency of FedHGNN [22] and FedGNN [17] on three datasets. Our observations from the data are as follows:

- When the learning rate is higher, models typically converge faster (i.e., reach the peak more quickly). however, slower convergence with lower learning rates often results in better performance (all models are obtained at a rate 0.01).

- It is observed that HGNN-based methods typically converge within 500-1000 steps, while FedGNN converges within 1000-2000 steps. This observation demonstrates that heterogeneous information is beneficial for reducing communication costs, as it provide more interactions for implicit feedback in each communication step.

- In Figure 8, compared to other models which converge within 500-1000 steps, our FedHGCL reaches convergence within 100-500 steps. Considering the sparsity of the Yelp dataset, we believe that contrastive learning (CL) plays a significant role in mitigating this issue. CL effectively aids local recommendation models in capturing user interest preferences through self-supervised signals generated by view alignment.

- In summary, fewer communication steps can reduce the number of communications between servers and clients, thereby lowering the risk of information leakage. Our FedHGCL demonstrates a significant advantage over the baseline in this regard, further confirming that rich and high-quality client data is crucial to federated recommendation.

5.5.5 Local Differential Privacy Parameters

This section discusses the impact of the parameters in the LDP module on model performance. The module's parameters are gradient clipping threshold δ

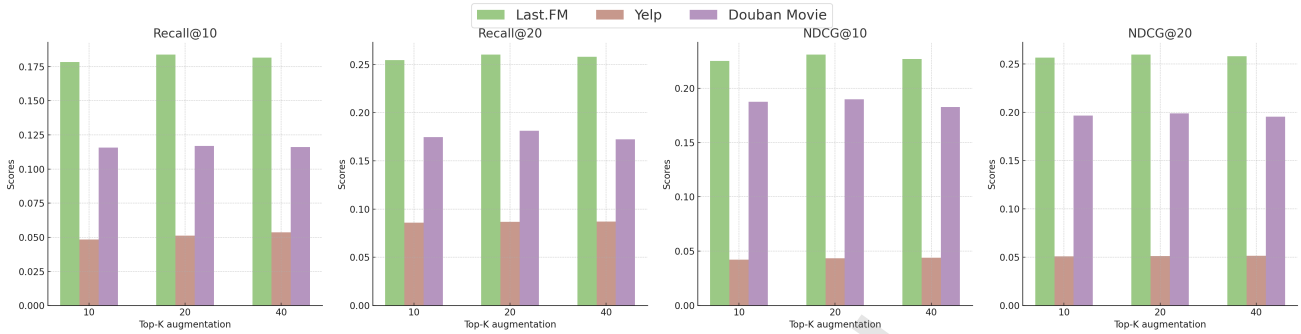
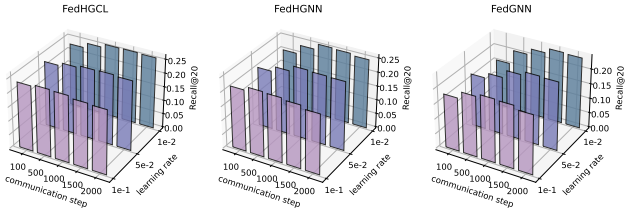
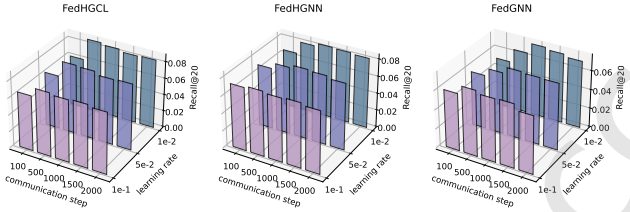


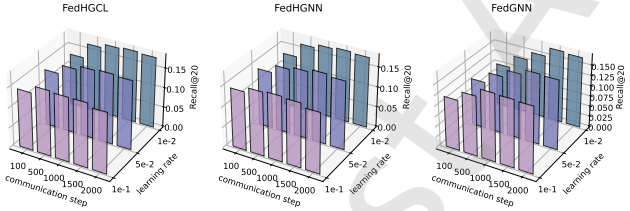
Fig. 7 Performance comparison of different data augmentation size on three datasets w.r.t. recall and NDCG. Best view in color.



(a) Last.FM

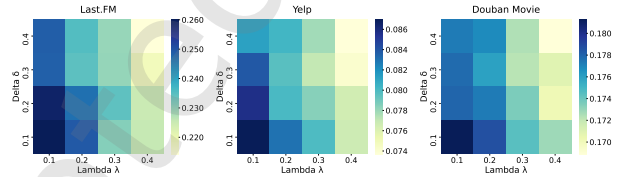


(b) Yelp

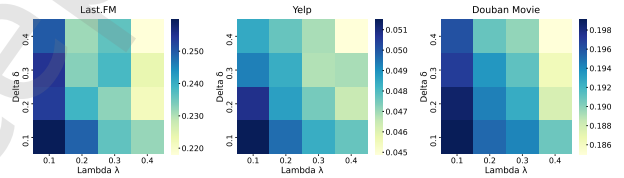


(c) Douban Movie

Fig. 8 Comparison of the three methods w.r.t. convergence efficiency on three datasets. Best view in color.



(a) Recall@20 w.r.t. LDP parameters λ and δ.



(b) NDCG@20 w.r.t. LDP parameters λ and δ.

Fig. 9 Performance comparison of the different gradient clipping threshold and Laplace noise on three datasets. Best view in color.

and Laplace noise λ (Equation 13). From Figure 9, it can be observed that across all datasets, as the clipping threshold δ and noise strength λ increase, there is a significant decrease in model performance. We believe the reasons for this trend are as follows:

- When the clipping threshold δ increases, more data points are considered noise and are clipped off, potentially leading to the loss of useful information.

- An increase in noise strength λ means a higher degree of random disturbances in the data, leading to a decline in data quality, and the model might learn noise instead of the actual signal.

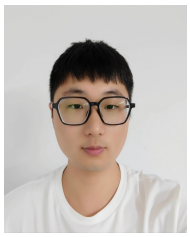
- Overall, LDP protects user privacy by injecting noise into gradients to alter them. Given that the server aggregates gradients containing noise, there is a declining trend in model performance.

6 CONCLUSION

In this paper, we first examined two persistent issues in existing federated recommender systems: performance degradation due to data sparsity and client generalization issues caused by heterogeneity. To address these limitations, we proposed FedHGCL, which utilized heterogeneous graphs and multi-view contrastive learning to effectively augment the user supervision signal. Firstly, we applied GNN encoding on semantic subgraphs based on varying meta-paths to obtain diverse embeddings for contrastive learning, a strategy aimed at overcoming the performance challenges posed by client data sparsity. Secondly,

we proposed an update and data augmentation strategy based on user sampling, averaging client data to mitigate the generalization issues caused by heterogeneity. Finally, our experiments with FedHGCL on three real-world datasets demonstrated the clear advantages of our approach. Even compared to some centralized learning methods, it achieved comparable or better results.

Yuan Wang is currently pursuing the Ph.D. degree with the School of Computer Science and Technology, Anhui University. His research interests include graph neural networks, recommender systems, and large language models.



Yu Wang received the M.S. degree from the School of Computer Science and Technology, Anhui University. His research has been published in leading journals and conferences, including IEEE TKDE, IEEE TBD, ACM TOIS, and ACM SIGIR. His current research interests include graph neural networks, recommender systems, and large language models.

Yiwen Zhang received the Ph.D. degree in management science and engineering from Hefei University of Technology, in 2013. He is currently a full professor with the School of Computer Science and Technology, Anhui University. He has published more than 100 papers in highly regarded conferences and journals, including IEEE TKDE, IEEE TMC, IEEE TSC, ACM TOIS, IEEE TPDS, IEEE TNNLS, ACM TKDD, SIGIR, ACL etc. His research interests include service computing, recommender systems,and big data analytics.

References

[1] F. Ricci, L. Rokach, and B. Shapira, "Introduction to recommender systems handbook," in *Recommender Systems Handbook*, pp. 1–35, Springer, 2010.

[2] L. Sang, Y. Wang, Y. Zhang, Y. Zhang, and X. Wu, "Intent-guided heterogeneous graph contrastive learning for recommendation," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 37, no. 4, pp. 1915–1929, 2025.

[3] M. Hou, L. Wu, Y. Liao, Y. Yang, Z. Zhang, C. Zheng, H. Wu, and R. Hong, "A survey on generative recommendation: Data, model, and tasks," *arXiv preprint arXiv:2510.27157*, 2025.

[4] T. Wu, W. Dou, F. Wu, S. Tang, C. Hu, and J. Chen, "A deployment optimization scheme over multimedia big data for large-scale media streaming application," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 12, no. 5s, pp. 1–23, 2016.

[5] L. Qi, X. Xu, X. Wu, Q. Ni, Y. Yuan, and X. Zhang, "Digital-twin-enabled 6g mobile network video streaming using mobile crowdsourcing," *IEEE Journal on Selected Areas in Communications*, vol. 41, no. 10, pp. 3161–3174, 2023.

[6] C. Shi, B. Hu, W. X. Zhao, and S. Y. Philip, "Heterogeneous information network embedding for recommendation," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 31, no. 2, pp. 357–370, 2018.

[7] J. Zhu, Y. Zhang, Y. Wang, W. Liao, R. Chen, and M. Yuan, "Knowledge-enhanced multi-task recommendation in hyperbolic space," *Applied Intelligence*, vol. 53, no. 23, pp. 28694–28710, 2023.

[8] M. Ammad-Ud-Din, E. Ivannikova, S. A. Khan, W. Oyomno, Q. Fu, K. E. Tan, and A. Flanagan, "Federated collaborative filtering for privacy-preserving personalized recommendation system," *CoRR abs/1901.09888*, 2019.

[9] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval (SIGIR)*, pp. 165–174, 2019.

[10] M. Wen, H. Mei, W. Wang, X. Xue, and X. Zhang, "Multi-task recommendation based on dynamic knowledge graph," *Applied Intelligence*, pp. 1–19, 2024.

[11] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[12] D. Chai, L. Wang, K. Chen, and Q. Yang, "Secure federated matrix factorization," *IEEE Intelligent Systems*, vol. 36, no. 5, pp. 11–20, 2020.

[13] L. Sang, Y. Wang, Y. Zhang, and X. Wu, "Denoising heterogeneous graph pre-training framework for recommendation," *ACM Transactions on Information Systems (TOIS)*, vol. 43, no. 5, pp. 1–31, 2025.

[14] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics (AISTATS)*, pp. 1273–1282, PMLR, 2017.

[15] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," *Knowledge-Based Systems (KBS)*, vol. 216, p. 106775, 2021.

[16] L. Qi, X. Zhang, W. Dou, C. Hu, C. Yang, and J. Chen, "A two-stage locality-sensitive hashing based approach for privacy-preserving mobile service recommendation in cross-platform edge environment," *Future Generation Computer Systems*, vol. 88, pp. 636–643, 2018.

[17] C. Wu, F. Wu, L. Lyu, T. Qi, Y. Huang, and X. Xie, "A federated graph neural network framework for privacy-preserving personalization," *Nature Communications*, vol. 13, no. 1, p. 3091, 2022.

[18] S. Luo, Y. Xiao, X. Zhang, Y. Liu, W. Ding, and L. Song, "Perfedrec++: Enhancing personalized federated recommendation with self-supervised pre-training," *arXiv preprint arXiv:2305.06622*, 2023.

[19] Z. Liu, L. Yang, Z. Fan, H. Peng, and P. S. Yu, "Federated social recommendation with graph neural network," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 13, no. 4, pp. 1–24, 2022.

[20] L. Luo and B. Liu, "Dual-contrastive for federated social recommendation," in *2022 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2022.

[21] Y. Wei, X. Fu, Q. Sun, H. Peng, J. Wu, J. Wang, and X. Li, "Heterogeneous graph neural network for privacy-preserving recommendation," in *2022 IEEE International Conference on Data Mining (ICDM)*, pp. 528–537, IEEE, 2022.

[22] B. Yan, Y. Cao, H. Wang, W. Yang, J. Du, and C. Shi, "Federated heterogeneous graph neural network for privacy-preserving recommendation," in *Proceedings of the ACM on Web Conference 2024 (WWW)*, pp. 3919–3929, 2024.

[23] C. Zhang, G. Long, T. Zhou, P. Yan, Z. Zhang, C. Zhang, and B. Yang, "Dual personalization on federated recommendation," in *Proceedings of the 32th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 4558–4566, 2023.

[24] Y. Wang, L. Sang, Y. Zhang, Y. Zhang, and X. Wu, "Generative-contrastive heterogeneous graph neural network," *IEEE Transactions on Big Data (TBD)*, 2025.

[25] J. Liu, H. Gao, C. Yang, C. Shi, T. Yang, H. Cheng, Q. Xie, X. Wang, and D. Wang, "Heterogeneous spatio-temporal graph contrastive learning for point-of-interest recommendation," *Tsinghua Science and Technology*, vol. 30, no. 1, pp. 186–197, 2024.

[26] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous graph attention network," in *The world wide web conference (WWW)*, pp. 2022–2032, 2019.

[27] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International Conference on Machine Learning (ICML)*, pp. 1597–1607, PMLR, 2020.

[28] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "Lightgcn: Simplifying and powering graph convolution network for recommendation," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 639–648, 2020.

[29] S. Luo, Y. Xiao, and L. Song, "Personalized federated recommendation via joint representation learning, user clustering, and model adaptation," in *Proceedings of the 31st ACM International Conference on Information & Knowledge Management (CIKM)*, pp. 4289–4293, 2022.

[30] S. Truex, L. Liu, K.-H. Chow, M. E. Gursoy, and W. Wei, "Ldp-fed: Federated learning with local differential privacy," in *Proceedings of the third ACM international workshop on edge systems, analytics and networking (EdgeSys)*, pp. 61–66, 2020.

[31] W.-S. Choi, M. Tomei, J. R. S. Vicarte, P. K. Hanumolu, and R. Kumar, "Guaranteeing local differential privacy on ultra-low-power systems," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, pp. 561–574, IEEE, 2018.

[32] F. Fei, S. Li, H. Dai, C. Hu, W. Dou, and Q. Ni, "A k-anonymity based schema for location privacy preservation," *IEEE Transactions on Sustainable Computing*, vol. 4, no. 2, pp. 156–167, 2017.

[33] C. He, K. Balasubramanian, E. Ceyani, C. Yang, H. Xie, L. Sun, L. He, L. Yang, P. S. Yu, Y. Rong, *et al.*, "Fedgraphnn: A federated learning system and benchmark for graph neural networks," *CoRR abs/2104.07145*, 2021.

[34] L. Qi, R. Wang, C. Hu, S. Li, Q. He, and X. Xu, "Time-aware distributed service recommendation with privacy-preservation," *Information Sciences*, vol. 480, pp. 354–364, 2019.

[35] H. Xu, M. Hou, L. Wu, F. Liu, Y. Yang, H. Bai, R. Hong, and M. Wang, "Fair Personalized Learner Modeling Without Sensitive Attributes," in *Proceedings of the ACM on Web Conference 2025*, pp. 4612–4624, 2025.

[36] Y. Zhang, Y. Zhang, D. Yan, S. Deng, and Y. Yang, "Revisiting graph-based recommender systems from the perspective of variational auto-encoder," *ACM Transactions on Information Systems (TOIS)*, vol. 41, no. 3, pp. 1–28, 2023.

[37] P. Guo, B. Ye, Y. Chen, T. Li, Y. Yang, X. Qian, and X. Yu, "A differential privacy protection protocol based on location entropy," *Tsinghua Science and Technology*, vol. 28, no. 3, pp. 452–463, 2022.

[38] M. Chen, C. Huang, L. Xia, W. Wei, Y. Xu, and R. Luo, "Heterogeneous graph contrastive learning for recommendation," in *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining (WSDM)*, pp. 544–552, 2023.

[39] J. Wu, X. Wang, F. Feng, X. He, L. Chen, J. Lian, and X. Xie, "Self-supervised graph learning for recommendation," in *Proceedings of the 44th international ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 726–735, 2021.

[40] X. Wang, N. Liu, H. Han, and C. Shi, "Self-supervised heterogeneous graph neural network with co-contrastive learning," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 1726–1736, 2021.

[41] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen, "Graph contrastive learning with augmentations," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 5812–5823, 2020.

[42] J. Wang, J. Wu, C. Jia, and Z. Zhang, "Self-supervised variational autoencoder towards recommendation by nested contrastive learning," *Applied Intelligence*, vol. 53, no. 15, pp. 18887–18897, 2023.

[43] F. Wang, L. Qi, W. Liu, B. Yu, J. Chen, and Y. Xu, "Inter-and intra-similarity preserved counterfactual incentive effect estimation for recommendation systems," *ACM Transactions on Information Systems (TOIS)*, vol. 43, no. 6, pp. 1–24, 2025.

[44] J. Yu, H. Yin, X. Xia, T. Chen, L. Cui, and Q. V. H. Nguyen, "Are graph augmentations necessary? simple graph contrastive learning for recommendation," in *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval (SIGIR)*, pp. 1294–1303, 2022.

[45] Y. Wang, Y. Yang, L. Wu, Y. Zhang, and R. Hong, "Multimodal Large Language Models with Adaptive Preference Optimization for Sequential Recommendation," *arXiv preprint arXiv:2511.18740*, 2025.

[46] Z. Lin, C. Tian, Y. Hou, and W. X. Zhao, "Improving graph collaborative filtering with

neighborhood-enriched contrastive learning," in *Proceedings of the ACM web conference 2022 (WWW)*, pp. 2320–2329, 2022.

[47] Y. Wang, L. Sang, Y. Zhang, and Y. Zhang, "Intent representation learning with large language model for recommendation," in *Proceedings of the 48th international ACM SIGIR conference on research and development in information retrieval (SIGIR)*, pp. 1870–1879, 2025.

[48] J. Wang, Y. Cao, F. Zhang, F. Kou, K. Wei, J. Zhang, and J. Chen, "IDBR: Interaction-Aware Dual-Granularity Learning for Bundle Recommendation," *Big Data Mining and Analytics*, vol. 8, no. 3, pp. 751–766, 2025.

[49] X. Cai, C. Huang, L. Xia, and X. Ren, "Lightgcl: Simple yet effective graph contrastive learning for recommendation," in *The Eleventh International Conference on Learning Representations (ICLR)*, 2023.

[50] C. Wang, Y. Yu, W. Ma, M. Zhang, C. Chen, Y. Liu, and S. Ma, "Towards representation alignment and uniformity in collaborative filtering," in *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining (KDD)*, pp. 1816–1825, 2022.

[51] J. Zhang and Q. Xu, "Attention-aware heterogeneous graph neural network," *Big Data Mining and Analytics*, vol. 4, no. 4, pp. 233–241, 2021.

[52] L. Sang, H. Yuan, Y. Huang, and Y. Zhang, "Graph structure learning for robust recommendation," *Tsinghua Science and Technology*, vol. 30, no. 4, pp. 1617–1635, 2024.

[53] K. Taha, P. D. Yoo, C. Yeun, and A. Taha, "Empirical and experimental perspectives on big data in recommendation systems: a comprehensive survey," *Big Data Mining and Analytics*, vol. 7, no. 3, pp. 964–1014, 2024.

[54] Y. Tian, K. Dong, C. Zhang, C. Zhang, and N. V. Chawla, "Heterogeneous graph masked autoencoders," in *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, vol. 37, pp. 9997–10005, 2023.

[55] J. D. M.-W. C. Kenton and L. K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of NAACL-HLT*, pp. 4171–4186, 2019.

[56] Z. Hou, X. Liu, Y. Cen, Y. Dong, H. Yang, C. Wang, and J. Tang, "Graphmae: Self-supervised masked graph autoencoders," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 594–604, 2022.

[57] Y. He, L. Meng, J. Ma, Y. Zhang, Q. Wu, W. Ding, and F. Yang, "Hierarchical bottleneck for heterogeneous graph representation," *Information Sciences*, vol. 667, p. 120422, 2024.

[58] H. Liu, N. Li, H. Kou, S. Meng, and Q. Li, "FSRPCL: privacy-preserve federated social relationship prediction with contrastive learning," *Tsinghua Science and Technology*, 2024.

[59] X. Qu, C. Guan, G. Xie, Z. Tian, K. Sood, C. Sun, and L. Cui, "Personalized federated learning for heterogeneous residential load forecasting," *Big Data Mining and Analytics*, vol. 6, no. 4, pp. 421–432, 2023.

[60] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," *arXiv preprint arXiv:1205.2618*, 2012.

[61] W. Dou, W. Tang, X. Wu, L. Qi, X. Xu, X. Zhang, and C. Hu, "An insurance theory based optimal cyber-insurance contract against moral hazard," *Information Sciences*, vol. 527, pp. 576–589, 2020.

[62] R. Gu, S. Wang, H. Dai, X. Chen, Z. Wang, W. Bao, J. Zheng, Y. Tu, Y. Huang, L. Qi, *et al.*, "Fluid-shuttle: Efficient cloud data transmission based on serverless computing compression," *IEEE/ACM Transactions on Networking (TON)*, 2024.

[63] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth International Conference on Artificial Intelligence and Statistics (ICAIS)*, pp. 249–256, JMLR, 2010.

[64] D. Kingma, "Adam: a method for stochastic optimization," in *Int Conf Learn Represent (ICLR)*, 2014.

[65] S. Rendle and C. Freudenthaler, "Improving pairwise learning for item recommendation from implicit feedback," in *Proceedings of the 7th ACM International Conference on Web Search and Data Mining (WSDM)*, pp. 273–282, 2014.