FedCIA: Federated Collaborative Information Aggregation for Privacy-Preserving Recommendation

Mingzhe Han
Fudan University
Shanghai, China
mzhan22@m.fudan.edu.cn

Dongsheng Li
Microsoft Research Asia
Shanghai, China
dongshengli@fudan.edu.cn

Jiafeng Xia
Fudan University
Shanghai, China
jfxia19@fudan.edu.cn

Jiahao Liu
Fudan University
Shanghai, China
jiahaoliu21@m.fudan.edu.cn

Hansu Gu
Seattle, United States
hansug@acm.org

Peng Zhang[*]
Fudan University
Shanghai, China
zhangpeng_@fudan.edu.cn

Ning Gu
Fudan University
Shanghai, China
ninggu@fudan.edu.cn

Tun Lu[*]
Fudan University
Shanghai, China
lutun@fudan.edu.cn

Abstract

Recommendation algorithms rely on user historical interactions to deliver personalized suggestions, which raises significant privacy concerns. Federated recommendation algorithms tackle this issue by combining local model training with server-side model aggregation, where most existing algorithms use a uniform weighted summation to aggregate item embeddings from different client models. This approach has three major limitations: 1) information loss during aggregation, 2) failure to retain personalized local features, and 3) incompatibility with parameter-free recommendation algorithms. To address these limitations, we first review the development of recommendation algorithms and recognize that their core function is to share collaborative information, specifically the global relationship between users and items. With this understanding, we propose a novel aggregation paradigm named collaborative information aggregation, which focuses on sharing collaborative information rather than item parameters. Based on this new paradigm, we introduce the federated collaborative information aggregation (FedCIA) method for privacy-preserving recommendation. This method requires each client to upload item similarity matrices for aggregation, which allows clients to align their local models without constraining embeddings to a unified vector space. As a result, it mitigates information loss caused by direct summation, preserves the personalized embedding distributions of individual clients, and supports the aggregation of parameter-free models. Theoretical analysis and experimental results on real-world datasets demonstrate the superior performance of FedCIA compared with the state-of-the-art federated recommendation algorithms. Code is available at https://github.com/Mingzhe-Han/FedCIA.

[*]Corresponding author.

CCS Concepts

• **Information systems → Recommender systems**; • **Security and privacy → Privacy protections**.

Keywords

recommendation, federated learning, collaborative information

ACM Reference Format:
Mingzhe Han, Dongsheng Li, Jiafeng Xia, Jiahao Liu, Hansu Gu, Peng Zhang, Ning Gu, and Tun Lu. 2025. FedCIA: Federated Collaborative Information Aggregation for Privacy-Preserving Recommendation. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '25), July 13–18, 2025, Padua, Italy.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3726302.3729977

1 Introduction

Recommendation algorithms [3, 17, 22, 28, 29] aim to recommend items that users may be interested in based on their historical interactions. However, these interactions are private and regulations such as GDPR [4] have been established to protect them, making it challenging for data holders to share information for unified model training. Federated recommendation algorithm [36, 37, 44] addresses this issue by treating each data holder (e.g., users or companies) as a client. It allows these clients to train models on their own devices and then upload the trained models to a central server for aggregation. This approach enables the development of a reliable recommendation model while protecting user privacy.

Recent federated recommendation algorithms [32, 47, 48] typically follow a unified aggregation paradigm. Each client initially trains their recommendation model based on their data. To protect user privacy, the embedding network is divided into user embeddings and item embeddings, where user embeddings are kept locally and item embeddings are uploaded and aggregated on a central

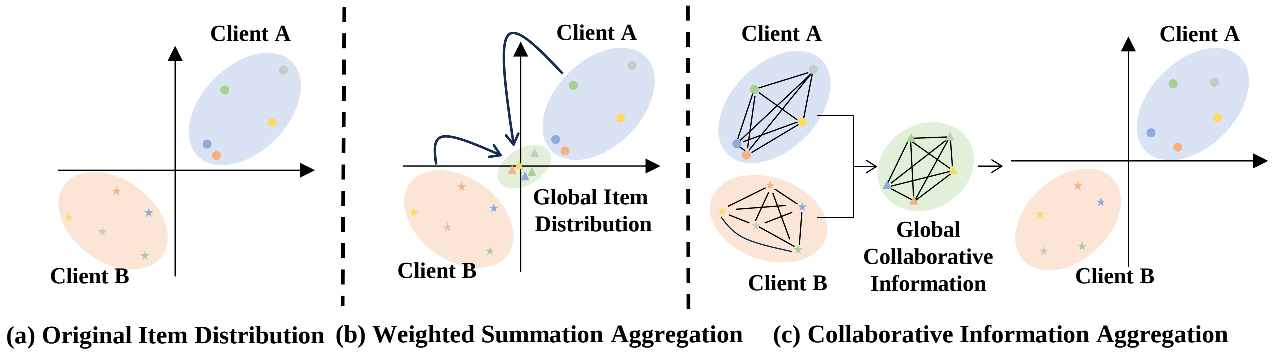
Client A
Client B
(a) Original Item Distribution
Client A
Client B
Global Item Distribution
(b) Weighted Summation Aggregation
Client A
Client B
Global Collaborative Information
Client A
Client B
(c) Collaborative Information Aggregation

Figure 1: Illustration of two aggregation paradigms in federated recommendation algorithms.

server by specific algorithms such as weighted average [31]. The aggregated item embeddings are then distributed back to each client for the next round of training. We refer to this aggregation paradigm as **Weighted Summation Aggregation**, which aligns item embeddings across different clients into a common vector space, enabling clients to leverage shared information effectively.

Although this aggregation algorithm has been widely successful in various federated recommendation algorithms, it presents three issues. **Firstly, the summation of item embeddings can lead to information loss.** For example, as illustrated in Figure 1, two clients, A and B, train their recommendation algorithm models with their own datasets, resulting in opposite item vector spaces. Directly summing these vectors will result in the aggregated server embeddings that tend to be zero-vectors, which lack useful information. **Secondly, there is a lack of personalized user modeling.** After client aggregation, the server distributes unified items to each client, ensuring all client items are in a consistent vector space. However, this global distribution space is not optimal for every client, it will result in a loss of personalized modeling for specific users, leading to suboptimal results. Some personalized federated recommendation systems [47, 48], achieve personalization through dual model and optimization. However, they still rely on item embeddings for aggregation, which can lead to reduced convergence efficiency and effectiveness due to information loss during aggregation. **Finally, they are unable to accommodate parameter-free Collaborative Filtering (CF) models, e.g. Graph Signal Processing.** These algorithms require that all clients use the parameter-based model. However, in real-world scenarios, the diversity of devices and user requirements often demands varied model structures, necessitating federated learning frameworks to have greater scalability.

In this paper, we suggest that the root cause of these challenges lies in the over-reliance on the model parameter during the aggregation. To address this issue, we revisit the evolution of recommendation algorithms over the years and observe a critical oversight. The superior performance of deep learning [7, 8, 30, 43] has encouraged researchers to parameterize users and items, while ignoring the collaborative information that played a fundamental role in early recommendation algorithms [9, 12]. Here the collaborative information indicates the relationship between all users and items. Therefore, when parameter aggregation within the federated learning framework results in information loss, it becomes essential to reconsider the importance of collaborative information. To this end, we propose a novel federated learning aggregation paradigm named **Collaborative Information Aggregation**. As illustrated in Figure 1, this paradigm no longer requires the summation of model parameters in federated learning. Instead, it focuses on sharing collaborative information (e.g. item similarity [33]) as modeled by their respective recommendation algorithms.

Based on this aggregation paradigm, we propose a novel federated recommendation algorithm framework named Federated Collaborative Information Aggregation (FedCIA). In this framework, we choose item similarity as the collaborative information in aggregation, thus each client uploads item similarity matrices instead of item embeddings. The algorithm aggregates these similarity matrices to derive a global item similarity matrix, which is then distributed to each client. Each client optimizes its local model to align its local item distribution with the global item. On one hand, as shown in Figure 1, this framework prevents information loss due to varying item distributions during aggregation. On the other hand, this framework avoids forcing item embeddings into a uniform vector space, thus preserving the personalized information of each client. In addition, this aggregation does not require parameter uploads, making it ideal for aggregation among parameter-free models. We also provide a theoretical analysis from the perspective of graph signal processing, demonstrating that aggregating item similarity optimally captures collaborative information.

Our main contributions are summarized as follows:

- We introduce a novel aggregation framework FedCIA for federated recommendation algorithms. To our knowledge, this is the first federated learning work that aggregates collaborative information instead of model parameters.
- FedCIA does not need to upload parameters, which can be utilized in parameter-free collaborative filtering algorithms. To our knowledge, this is the first federated learning work compatible with parameter-free recommendation algorithms.
- We propose to leverage the sum of similarity matrices to aggregate collaborative information and provide a theoretical analysis to support this algorithm.
- Experimental results on real-world datasets demonstrate that FedCIA achieves higher accuracy compared to existing state-of-the-art federated recommendation algorithms.

2 Related Work

2.1 Federated Learning

Federated Learning (FL) [18, 19, 45] is a distributed framework enabling multiple clients to collaboratively train a central model while retaining their raw data locally. The foundational algorithm, FedAvg [31], aggregates client models by weighted summing them based on the number of samples in each dataset. Subsequent research has primarily focused on server-side aggregation and client-side training. For instance, FedAvgm [11] introduces a momentum module that incorporates previous aggregation rounds to enhance model updates. FedProx [20] constrains parameter changes during each client's training round by using a loss function. These studies have not specifically addressed recommendation algorithms, where data distribution is more dispersed and certain user-related parameters pose privacy concerns.

2.2 Federated recommendation algorithms

The federated recommendation algorithm [32, 36, 37, 44, 47, 48] integrates the federated learning framework into recommendation models while incorporating specific designs tailored to the unique requirements of these systems. FedRec [21] mitigates privacy risks by randomly sampling unrated items and assigning virtual ratings. This approach prevents servers from inferring user interactions based on gradient information. FedMF [1] employs homomorphic encryption during the aggregation process, ensuring that user ratings remain private while allowing the server to aggregate gradients securely. FedNCF [32] extended this work by applying the commonly used Neural Collaborative Filtering (NCF) framework, partitioning parameters into user-related and item-related components. Only the item-related parameters are uploaded during aggregation. While these approaches successfully achieve model aggregation while safeguarding user privacy, they impose a uniform parameter structure across all clients, disregarding individual user preferences. This limitation often leads to suboptimal solutions by failing to account for personalized information.

Recognizing the importance of personalization in federated recommendation algorithms, recent research has explored personalized federated learning approaches. PFedRec [47] introduces a dual personalization mechanism that optimizes both the global model and individual client models to provide user-specific recommendations. GPFedRec [48] takes a different approach by constructing a user relationship graph to capture inter-user correlations and leveraging local embeddings to model personalized features. However, they still rely on weighted summation of item features during aggregation. This aggregation strategy can lead to the loss of collaborative information, ultimately reducing the quality of the final model. In contrast, our method mitigates this issue by preserving collaborative information during aggregation, ensuring more robust and accurate recommendations.

3 Preliminaries

3.1 Recommendation Algorithms

Recommendation algorithms aim to predict the user-item interaction y between a user u and an item i based on the user history dataset $\mathcal{D} = (\mathcal{U}, \mathcal{I}, \mathcal{Y})$, where $\mathcal{U}$ represents the set of users, $\mathcal{I}$ represents the set of items, and $\mathcal{Y}$ represents the interaction labels. The label y is a binary variable ($y \in \{0, 1\}$) indicating whether the user u is interested in the item i ($y = 1$) or not ($y = 0$).

Most recommendation algorithms $R(.)$ follow the same paradigm. First, the user u and the item i are mapped into embeddings through embedding networks $E_u(.|\theta_u)$ and $E_i(.|\theta_i)$, respectively. Then the embeddings are used to compute a prediction score $\hat{y}$ through a score network $S(.|\theta_s)$. Formally, the recommendation algorithm can be expressed as $\hat{y} = R(u, i) = S(E(u, i))$ where $E(u, i)$ represents the combined embeddings of the user and item, and $S(.)$ is the function used to generate the prediction score. The model parameters θ of the recommendation algorithm $R(.)$ are divided into three components: the user embedding parameters θ_u, the item embedding parameters θ_i and the score network parameters θ_s. In some recommendation algorithms, the prediction score is obtained directly from the dot product of the user and item embeddings. In such cases, the score network θ_s does not include any additional parameters, simplifying the model architecture.

3.2 Federated Recommendation Algorithms

In federated learning, users store their interaction history on their own devices to ensure privacy. We assume that there are K user clusters, each containing one or more users. Thus, the dataset for federated learning consists of K sets of data samples distributed across K clients, denoted as $\mathcal{D}_k = (\mathcal{U}, \mathcal{I}, \mathcal{Y})_k$, where $k \in [1, K]$. The goal of federated learning is to train a global model $R(.)$ while protecting the interaction data of each user. In conventional federated learning, such as FedAvg [31], each client first trains its local model based on its local dataset $\mathcal{D}_k$, The model parameters are then uploaded to a server for aggregation using a weighted average algorithm, based on the number of samples in each dataset: $R(.|\theta) = \sum_{k=1}^{K} \frac{|\mathcal{D}_k|}{|\mathcal{D}|} R_k(.|\theta)$.

In recommendation algorithms, due to the unique properties of parameters, a different aggregation algorithm is used. User embeddings θ_u are directly linked to individual users, and clients keep these embeddings local to maintain privacy. Instead, upload and aggregate other parameters θ/θ_u. This approach facilitates information sharing among clients while preserving user privacy, effectively training a global federated recommendation algorithm model.

3.3 Graph Signal Processing

Given a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ comprising N nodes, it can be represented by an adjacency matrix A, where A_{ij} indicates the presence of a relationship between nodes i and j. The normalized form of the adjacency matrix is denoted as $\tilde{A} = D^{1/2}AD^{-1/2}$, where D is the degree matrix. In the field of graph signal processing, filters are crucial tools used to uncover complex relationships between nodes and accurately predict missing links. Filters are primarily designed based on the (normalized) Laplacian matrix [2, 35] $\tilde{L} = I - \tilde{A}$, expressed as: $F = f(\tilde{L}) = f(U\Sigma U^\top) = U\text{Diag}([f(\lambda_1), \cdots, f(\lambda_N)])U^T$ where $f(.)$ represents the frequency response function, U and Σ denote the eigenvector matrix and eigenvalue matrix of $\tilde{L}$, respectively. Different frequency response functions can construct various filters, thereby uncovering different relationships between nodes.

Graph signal processing is widely applied in recommendation systems[23–27, 40–42] for its significant effectiveness and efficiency.

In recommendation algorithms, users and items are represented as nodes in a bipartite graph, and their interactions are modeled as edges. Generally, the graph is formally represented using an adjacency matrix $A \in \{0,1\}^{N\times M}$, where N is the number of users and M is the number of items. Each entry $A_{ij} = 1$ indicates that user i has interacted with item j; otherwise, $A_{ij} = 0$.

The filter F in graph signal processing methods is typically an $M\times M$ matrix that encodes the relationships or correlations between items. The simplest filter that those methods adopt is the linear filter, which aggregates the interaction matrix after normalization. Specifically, the filter can be expressed as $F = \tilde{A}^T\tilde{A}$, where $\tilde{A}$ is the normalized interaction matrix. The normalization process is defined as $\tilde{A} = D_u^{-1/2}AD_i^{-1/2}$, where D_u is the degree matrix of the users and D_i is the degree matrix of the items.

4 Theoretical Analysis

In this section, we first propose that the item similarity in parameter-based models can be interpreted as a linear filter within the framework of graph signal processing, a parameter-free algorithm. This enables the application of theoretical analysis from graph signal processing to both parameter-based and parameter-free collaborative filtering models. We then emphasize that the average of item similarities can approximate global item similarity. Finally, we identify the limitations of the existing weighted summation aggregation.

4.1 Item Similarity and Linear Filters

In parameter-based recommendation algorithm methods, items are mapped to item embeddings, which encapsulate the model's representation of item features. Intuitively, the similarity between items, computed as the dot product of their corresponding item embeddings, can be interpreted as a linear filter. Taking Matrix Factorization (MF) as an example, it decomposes the interaction matrix into user embeddings and item embeddings and optimizes these embeddings to find the optimal solution by minimizing a loss function. One of the optimal solutions can be understood through Singular Value Decomposition (SVD). Specifically, SVD decomposes the interaction matrix A into $U\Sigma_{\tilde{A}}V^T$, where $\sqrt{\Sigma_{\tilde{A}}}V^T$ can be viewed as the item embeddings optimized by MF. The dot product of these item embeddings can then be expressed as $V^T\Sigma_{\tilde{A}}V$, which represents a linear filter whose frequency response function is $f(\lambda) = 1 - \lambda$ in this context.

4.2 Collaborative Information Aggregation

We begin by analyzing the aggregation of collaborative information, with a particular emphasis on the impact of privacy considerations in this process. We analyzed centralized learning without considering privacy and federated learning with privacy considerations and proposed the following theorem.

Theorem 1. *In a federated recommendation algorithm, assuming that the popularity of each item (i.e., the number of user interactions) is identical, the average of each local linear filter is equivalent to the global ideal linear filter.*

Note that the 'global ideal linear filter' here refers to a global linear filter that does not consider privacy risks, not an 'ideal filter'. And here is the proof.

Proof. Consider the complete global interaction matrix as $A_s \in \{0,1\}^{N\times M}$. In an ideal scenario where no privacy risks are considered, complete user data is processed on a central server, and the linear filter is:

$$F_{ideal} = \tilde{A_s}^T\tilde{A_s} = D_i^{-1/2}A_sD_u^{-1}A_sD_i^{-1/2}. \tag{1}$$

In reality, due to privacy risks, data is stored on each client. The interaction matrix is split among N clients, where client k holds $A_k \in \{0,1\}^{1\times M}$. Each client generates a local linear filter $F_k = \tilde{A_k^T}\tilde{A_k}$ using its interaction matrix. The average of these N filters are:

$$\begin{aligned}F_{agg} &= \frac{1}{N}(F_1 + F_2 + \ldots + F_N)\\ &= \frac{1}{N}(D_{i1}^{-1/2}A_1D_{u1}^{-1}A_1D_{i1}^{-1/2} + D_{i2}^{-1/2}A_2D_{u2}^{-1}A_2D_{i2}^{-1/2} + \ldots\\ &+ D_{iN}^{-1/2}A_ND_{uN}^{-1}A_ND_{iN}^{-1/2})\\ &= \frac{1}{N}(A_1D_{u1}^{-1}A_1 + A_2D_{u2}^{-1}A_2 + \ldots + A_ND_{uN}^{-1}A_N)\\ &= \frac{1}{N}[A_1||A_2||\ldots||AN]^T[D_{u1}||D_{u2}||\ldots||D_{uN}][A_1||A_2||\ldots||AN]\\ &= \frac{1}{N}A_sD_u^{-1}A_s.\end{aligned} \tag{2}$$

In federated learning, counting the exact number of interactions per item D_i is challenging due to privacy concerns. Assuming each item is interacted with once, we set $D_i = NI$. Thus, the ideal linear filter becomes:

$$F_s = D_i^{-1/2}A_sD_u^{-1}A_sD_i^{-1/2} = \frac{1}{N}A_sD_u^{-1}A_s = F_{agg}. \tag{3}$$

This shows that under the assumption of equal item popularity, the average of local linear filters equals the global ideal linear filter. □

Therefore, by considering item similarity as a filter in graph signal processing, the global filter can approximate the mean of multiple local filters. This approach enables the aggregation of item similarity matrices from individual clients to approximate a global item similarity matrix.

4.3 Loss Information during Aggregation

In the previous section, we demonstrated that item similarity serves as an effective medium for information aggregation. In this section, we analyze the impact of different aggregation paradigms on global information. For simplicity and consistency, we adopt the L1 distance as the similarity metric between items, as it is used in clustering [14, 46] and metric learning [39]. The L1 distance, where larger values indicate weaker correlations, is used here to evaluate the amount of information captured by the correlation matrix. Since the L1 distance is always a positive scalar, we posit that a correlation matrix with a broader range of values conveys more information.

Consider two clients, A and B, each holding the embeddings of two items. Let the embeddings be e_{a1}, e_{a2} for client A and e_{b1}, e_{b2} for client B. In weighted summation aggregation, the global embeddings are obtained by directly summing the embeddings from

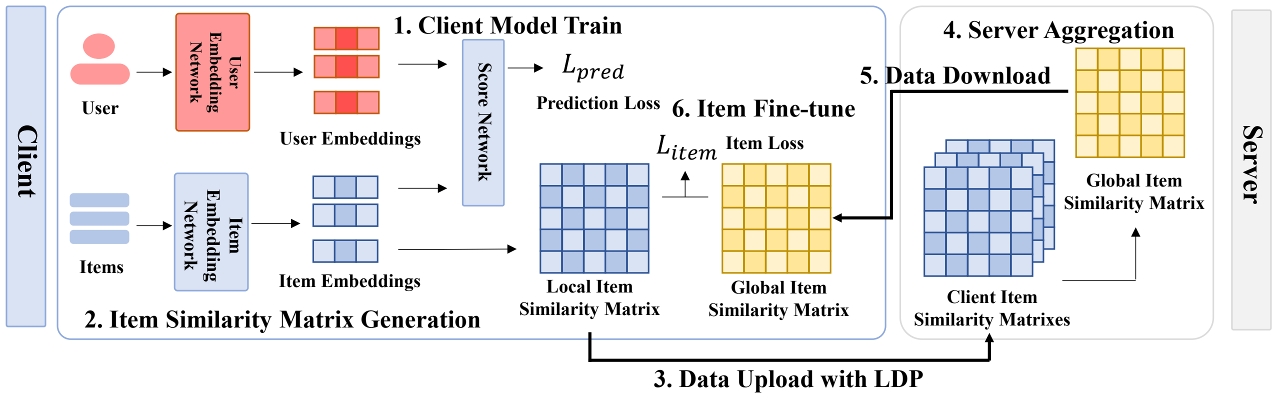

Figure 2: The illustration of the FedCIA framework. We only illustrate a single client for simplicity. The red part indicates private information and the yellow part indicates global information.

both clients. The recommendation algorithm then computes the similarity between items as $S_{WS} = \frac{1}{2}|(e_{a1}+e_{b1})-(e_{a2}+e_{b2})|$.

In contrast, collaborative information aggregation first computes item similarities locally before aggregation. By the triangle inequality, we have:

$$\begin{aligned} S_{CI} &= \frac{1}{2}|e_{a1}-e_{a2}| + \frac{1}{2}|e_{b1}-e_{b2}| \\ &\geq \frac{1}{2}|e_{a1}-e_{a2}+e_{b1}-e_{b2}| \qquad (4) \\ &= \frac{1}{2}|(e_{a1}+e_{b1})-(e_{a2}+e_{b2})| = S_{WS}. \end{aligned}$$

This inequality highlights that weighted summation aggregation can result in information loss, particularly when the embeddings from clients A and B have opposite values. In such cases, weighted summation aggregation fails to retain meaningful information. By contrast, collaborative information aggregation consistently preserves more information due to its larger value range, making it a more robust approach for constructing the global collaborative information.

5 Methodology

In this section, we introduce the proposed FedCIA framework, which is versatile and applicable to various recommendation algorithms, including both parameter-based and parameter-free collaborative filtering models. We first outline the algorithm for parameter-based models, followed by alternative solutions for parameter-free models. Finally, we discuss the challenges our method addresses in practical deployment.

5.1 Overview

We first introduce the process of our framework under parameter-based models. Figure 2 illustrates the complete process of FedCIA during a round of federated learning communication. Each client initially trains its model using local data. As described in Section 4, we use item similarity as the collaborative information for aggregation, and these terms (i.e. "collaborative information" and "item similarity") will be used interchangeably in this section. The local model then generates a local item similarity matrix based on the trained item embeddings, which is uploaded to the server. The server aggregates these matrices to form a global item similarity matrix. Finally, the client fine-tunes its item embeddings based on this global matrix.

5.2 Training Pipeline

5.2.1 Data Training. In the beginning, each client trains a recommendation model locally using its own data, formulated as:

$$L_{pred} = \frac{1}{|\mathcal{D}|} \sum_{(u,i,y)\in(\mathcal{U},\mathcal{I},\mathcal{Y})} L(\theta; R(u,i), y), \qquad (5)$$

where we use BPR loss as the loss function.

5.2.2 Data Upload. As described in Section 3, the recommendation algorithm model maps discrete item IDs to corresponding item embeddings. Based on these embeddings, the client computes the complete item similarity matrix $C = E(i|\theta_i)^T E(i|\theta_i)$ using a dot product. Each client k then uploads its local similarity matrix C_k as collaborative information to the server for aggregation.

5.2.3 Data Aggregation. Upon receiving the item similarity matrices from all clients, the server aggregates them to generate the global similarity matrix. In Section 4, we demonstrate that the average of local linear filters approximates the global linear filters. Leveraging this theorem, FedCIA aggregates the similarity matrices by taking the average of those uploaded by each client. Thus, the global similarity matrix at the server is: $C_s = \frac{1}{N}\sum_{k=1}^{K} C_k$.

5.2.4 Data Download. Our data downloads differ from weighted summation aggregation, where each local client can directly substitute its local item embeddings with global item embeddings from the server to incorporate aggregated information. However, the parameters of the local model typically do not include collaborative information, such as the item similarity matrix, which is not explicitly part of most recommendation algorithms. Although we obtain the global item similarity matrix through aggregation algorithms, this matrix cannot directly replace the local model's parameters. Instead, we must approximate the local model's collaborative information to align with the global collaborative information.

Intuitively, item similarity is derived from their embeddings, so the global item similarity can be treated as a label to fine-tune the local item embeddings, ensuring that the local item similarity matrix approximates the global item similarity matrix. We choose the Mean Squared Error (MSE) as the loss function for this fine-tuning process. The loss for the entire fine-tuning process is expressed as:

$$L_{item} = L(\theta_i; C_s) = \mathrm{MSE}(E(i|\theta_i)^T E(i|\theta_i), C_s). \tag{6}$$

5.2.5 Privacy Protection. In weighted summation aggregation methods, the privacy risks associated with uploading item embeddings were addressed by employing a Local Differential Privacy (LDP) strategy, where zero-mean Laplacian noise was added directly to the item embeddings. In our algorithm, we calculate the similarity matrix by dot products based on the item embeddings. This similarity matrix transmits less information because attackers can only infer item similarities but cannot directly recover the original item embeddings, thereby reducing the risk of information leakage. Nevertheless, we also apply zero-mean Laplacian noise to the similarity matrix before uploading it to mitigate the potential risk as: $C_i = C_i + Laplacian(0, \delta)$, where δ denotes the intensity of the Laplacian noise, which is derived from the privacy budget and the sensitivity of the similarity matrix.

5.2.6 Training Algorithm. We train the recommendation algorithm model through multiple rounds of communication. From the client's perspective, it initially trains a local recommendation algorithm model using its own data. It then generates the item similarity matrix locally based on this model. Finally, the client uploads the similarity matrix, enhanced with local differential privacy, to the server. On the server side, it aggregates the received item similarity matrices into a global similarity matrix by average. The complete model training pipeline is detailed in Algorithm 1.

Algorithm 1 FedCIA Training Pipeline

Input: Training set D_k for each client k.
Parameters: Maximum number of communication rounds com_num, maximum number of client item training epoch $epoch_i$, maximum number of client training epoch $epoch_c$, and the number of clients K.

```
1:  for $t = 1, \ldots, com\_num$ do
2:    Server aggregation:
3:    Generate the global item similarity matrix $C_s = \frac{1}{N}\sum_{k=1}^{K} C_k$.
4:    Local training:
5:    for $k = 1, \ldots, K$ do
6:      for $epoch = 1, \ldots, epoch_c$ do
7:        for $(u, i, y)$ in $(\mathcal{U}, \mathcal{I}, \mathcal{Y})$ do
8:          Train client model by $L_{pred}$.
9:        end for
10:     end for
11:     Generate the item similarity matrix $C_k$.
12:     Add Laplacian noise on item similarity matrix $C_k$.
13:     Upload the item similarity matrix to the server.
14:     Wait for Server aggregation.
15:     Downloads global similarity matrix $C_s$.
16:     for $epoch = 1, \ldots, epoch_i$ do
17:       Train client item embeddings by $L_{item}$.
18:     end for
19:   end for
20: end for
```

5.3 Parameter-free Model Aggregation

In this section, we extend our aggregation method to parameter-free collaborative filtering models. Existing weighted summation aggregation techniques rely on parameter calculations, making them unsuitable for parameter-free models like graph signal processing [34]. In contrast, FedCIA can perform aggregation without requiring model parameters, allowing it to be adapted for use with parameter-free models.

5.3.1 Collaborative Information Aggregation. In graph signal processing, the original interaction matrix is processed using various filters, and recommendations are generated based on the score rankings of the filtered results. To facilitate aggregation, we directly select the filter as the collaborative information to be shared. For algorithms that require a combination of multiple filters (e.g., GF-CF), we use the final mixed filter as the collaborative information to upload. Once the filters are uploaded, the server aggregates them into a global filter using an average approach, similar to the aggregation process for parameter-based models.

5.3.2 Collaborative Information Utilization. In graph signal processing, global collaborative information corresponds to a global filter F_{agg}, which each client can use to generate the final predicted interaction matrix. However, similar to parameter-based models, relying solely on global information may lead to a loss of personalized information. Therefore, we incorporate both the global filter and the individual filter F_k of each client k, as $\hat{A_k} = A_k F_k + \beta A_k F_{agg}$, where $\hat{A_i}$ refers to the prediction of interaction matrix A_i and β is a hyperparameter.

5.4 Discussion

In this section, we discuss the practical deployment of our FedCIA, focusing on model scalability and communication resources.

5.4.1 Model Scalability. In real-world federated recommendation systems, user devices often have varying computing capabilities, leading to models with different architectures. This diversity poses a challenge to the scalability of federated learning algorithms. Traditional parameter-based weighted summation aggregation requires consistent model architectures across clients, as differences in model size result in incompatible parameter vectors. Our proposed FedCIA addresses this issue by using collaborative information for aggregation, relying only on the similarity between different items. This similarity is typically obtained through the dot product of item embeddings. Consequently, FedCIA is applicable to all parameter-based recommendation models that encode items into embeddings. In our experiments, we evaluated the aggregation results under heterogeneous model conditions. The findings demonstrate that our method effectively aggregates information across different model architectures and sizes, exhibiting enhanced scalability.

Table 1: The overall comparison for all baseline methods in five datasets. The boldface indicates the best result and the underline indicates the secondary.

Dataset	Metric	No Aggregation			Weighted Summation Aggregation					Collaborative Information Aggregation (Ours)		
		MF	LightGCN	GF-CF	FedMF	FedNCF	PFedrec	GPFedrec	FedLightGCN	FedCIA$_{\text{MF}}$	FedCIA$_{\text{LightGCN}}$	FedCIA$_{\text{GF-CF}}$
Ml-100k	F1@10	0.0644	0.1012	0.1040	0.2074	0.1321	0.1428	0.1413	0.1828	0.2326	0.1881	**0.2356**
	MRR@10	0.2557	0.3760	0.3635	0.5578	0.4350	0.4750	0.4685	0.5653	**0.6008**	0.5601	0.5806
	NDCG@10	0.3307	0.4558	0.4515	0.6460	0.5244	0.5577	0.5555	0.6284	**0.6766**	0.6298	0.6704
Ml-1m	F1@10	0.0471	0.0951	0.1264	0.1106	0.0951	0.0989	0.0964	0.1690	0.1814	0.1687	**0.1845**
	MRR@10	0.1826	0.3043	0.3797	0.3367	0.3098	0.3387	0.3362	0.4470	0.4705	0.4527	**0.4815**
	NDCG@10	0.2490	0.3940	0.4667	0.4203	0.3941	0.4125	0.4072	0.5368	0.5561	0.5403	**0.5647**
Beauty	F1@10	0.0068	0.0222	0.0248	0.0068	0.0061	0.0182	0.0173	0.0245	0.0080	0.0250	**0.0358**
	MRR@10	0.0117	0.0355	0.0402	0.0123	0.0109	0.0349	0.0325	0.0397	0.0135	0.0404	**0.0571**
	NDCG@10	0.0172	0.0511	0.0576	0.0184	0.0165	0.0506	0.0478	0.0571	0.0206	0.0586	**0.0821**
LastFM	F1@10	0.0140	0.0315	0.0388	0.0398	0.0444	0.0439	0.0475	0.0606	0.0578	**0.0761**	0.0756
	MRR@10	0.2094	0.3003	0.3597	0.3408	0.4014	0.3666	0.4168	0.4792	0.4563	0.5272	**0.5466**
	NDCG@10	0.2691	0.3859	0.4503	0.4337	0.4866	0.4617	0.5007	0.5670	0.5385	0.6172	**0.6235**
BX	F1@10	0.0038	0.0050	0.0069	0.0102	0.0122	0.0137	0.0123	0.0107	0.0115	0.0140	**0.0331**
	MRR@10	0.0092	0.0119	0.0140	0.0217	0.0251	0.0296	0.0274	0.0233	0.0242	0.0290	**0.0625**
	NDCG@10	0.0136	0.0171	0.0202	0.0321	0.0374	0.0426	0.0393	0.0337	0.0358	0.0422	**0.0885**

Table 2: The statistics of our datasets.

datasets	# Users	# Items	# Interactions
Ml-100k	943	1682	100000
Ml-1m	6040	3706	1000209
Beauty (Amazon Beauty)	22363	12101	198502
LastFM	992	10000	517817
BX (Book-Crossing)	18964	19998	482153

5.4.2 Communication Resources. Our method inherently requires more communication resources. Rather than uploading model parameters, our approach involves uploading the item similarity matrix, sized at $R^{M \times M}$. In contrast, item embeddings are sized at $R^{M \times L_E}$, where L_E denotes the embedding dimension. Consequently, uploading the item similarity matrix directly often demands significantly more communication resources. However, in practice, a single training session typically does not utilize all items, and the item similarity matrix usually contains substantial noise. To mitigate communication overhead, we propose using truncated singular value decomposition to decompose the item similarity matrix, retaining only the top-L_k singular values and corresponding vectors. Given that the similarity matrix is symmetric, the left singular vectors are identical to the right singular vectors, so only half need to be transmitted. The server then reconstructs the filters for each client using these vectors and calculates the global similarity matrix. This method reduces the size of the uploaded data from the original $R^{M \times M}$ to $R^{M \times L_k} + R^{L_k}$, here L_k, similar to L_E, is approximately one percent of the item number M. This approach achieves a level of communication efficiency comparable to uploading model parameters while preserving most of the crucial information.

6 Experiments

In this section, we introduce and analyze the following research questions (RQs):

- **RQ1:** Does FedCIA outperform current state-of-the-art federated recommendation methods?
- **RQ2:** Does FedCIA mitigate the information loss and preserve the personalized embedding distributions?
- **RQ3:** Does FedCIA have better scalability across different model architectures?

6.1 Experimental Settings

6.1.1 Datasets. Our experiments are conducted on five widely used datasets: Ml-100k [5], Ml-1m [5], Beauty [6], Book-Crossing (BX)[49] and LastFM [16]. These datasets are split into training and test sets in an 8:2 ratio, with 10% of the training set used as a validation set. We divided each dataset into 100 clients for federated learning. Considering that some federated recommendation system frameworks are designed with a separate client for each user, we also considered this scenario in the Ml-100k dataset. The statistics of these datasets are shown in Table 2.

6.1.2 Compared Methods. We compared various baseline methods in our experiments. For the independent recommendation algorithm without aggregation, we select three backbone models: MF [15], LightGCN [7] and GF-CF [34]. For the federated recommendation algorithm, we select three popular methods: FedMF [1], FedLightGCN, FedNCF [32] and two methods that focus on user personalization: Pfedrec [47] and GPFedRec [48].

6.1.3 Evaluation Metrics. We use F1 [10], Mean Reciprocal Rank (MRR) [38] and Normalized Discounted Cumulative Gain (NDCG) [13] as evaluation metrics in our experiments (higher values indicate better accuracy). These are popular metrics in the Top-K recommendation scenario. We set $K = 10$ for these metrics.

6.1.4 Implementation Details. As we introduced in Section 5.4.1, FedCIA has strong scalability and can be applied to both parameter-based and parameter-free methods. To demonstrate this advantage, we applied our method to a simple recommendation system (MF), a graph-based recommendation system (LightGCN), and a parameter-free recommendation system (GF-CF), presenting all experimental results. We also select MLP as the model for a fair comparison with GPFedrec in RQ2. Our method involves generating a similarity matrix based on item embeddings. For MF and MLP, the item

Table 3: The comparison for different aggregations in Ml-100k dataset.

Backbone	Method	F1@10	MRR@10	NDCG@10
MLP	IM	0.0112	0.0748	0.1074
	WAA	0.0993	0.4176	0.4806
	GGA	0.1509	0.5005	0.5748
	CIA (ours)	**0.2176**	**0.5887**	**0.6637**
MF	IM	0.0108	0.0811	0.1151
	WAA	0.1105	0.4330	0.4986
	GGA	0.1203	0.4832	0.5445
	CIA (ours)	**0.2246**	**0.5782**	**0.6593**

embeddings can be directly used to calculate item similarity. In the case of LightGCN, there are two types of item embeddings: before and after convolution. To avoid the influence of different client graph structures on aggregation, we use the item embeddings before convolution for the calculation.

The implementation of FedCIA utilizes the PyTorch library on an NVIDIA Tesla T4 GPU. We maintain the same parameter range as FedCIA and its corresponding independent recommendation algorithm for a fair comparison. Specific parameter details can be found in the reproduced code.

6.2 Method Comparison (RQ1)

We compare FedCIA with all baseline methods in Table 1 and observe the following:

1) Both weighted summation aggregation methods and our collaborative information methods outperform the independent training algorithm. This suggests that, in the selected federated recommendation system dataset, information transfer between clients is crucial for better results. 2) Our method consistently outperforms weighted summation aggregation algorithms with the same backbone across all datasets. This improvement is due to our approach's reliance on collaborative information aggregation, which is more effective than parameter aggregation. 3) Pfedrec and GPfedrec address the issue of insufficient personalization caused by diverse client users in federated recommendation algorithms. However, they only consider aggregation at the individual user level, where each client lacks the user relationship graph. Consequently, graph-structured models like LightGCN and GF-CF do not perform effectively. In contrast, our approach extends federated recommendation scenarios to different user groups or companies, where each client can hold multiple users, enhancing the effectiveness of graph-based methods. Additionally, their methods aggregate parameters rather than collaborative information, resulting in inferior performance compared to ours. For a fair comparison, we conduct experiments for a single-user aggregation scenario in the next section (RQ2). 4) Existing federated learning frameworks cannot be directly applied to parameter-free models like graph signal processing models. However, FedCIA overcomes this limitation by effectively aggregating collaborative information and leveraging the superior performance of graph signal processing in handling graph structures. This enables it to achieve results comparable to those of parameter-based federated recommendation systems.

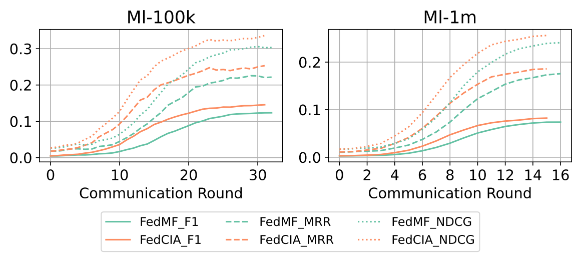

Figure 3: The learning curve for different aggregations on MF in Ml-100k and Ml-1m datasets.

6.3 Aggregation Comparison (RQ1)

The core algorithm proposed in FedCIA is the collaborative information aggregation algorithm, which is an independent module and is challenging to decompose for ablation studies. Instead, we compare our collaborative information aggregation algorithm with other aggregation methods under identical conditions to demonstrate its superiority. To ensure a fair comparison with personalized federated learning algorithms, we select MLP as the backbone model. Additionally, we configure each client to hold only one user, which accentuates the dispersed data and non-IID challenges, thereby highlighting the benefits of personalized federated learning. We evaluate the following aggregation algorithms:

- Independent Model (IM): Each client trains its own model independently without aggregation.
- Weighted Average Aggregation (WAA): Clients aggregate parameters by weighted summation based on their dataset size (e.g., FedAvg [31]).
- Graph-Guided Aggregation (GGA): Clients aggregate parameters by forming a relationship graph based on item similarity (e.g., GPFedrec [48]).
- Collaborative Information Aggregation (CIA): Our algorithm.

We conducted experiments on the Ml-100k dataset, and the results are presented in Table 3. The results indicate that our collaborative information aggregation consistently outperforms other methods even if we constrain them to the same backbone model. The Independent Model, lacking any aggregation, performs the worst. Weighted Average Aggregation, a classic federated learning approach, improves performance through parameter aggregation. Graph-Guided Aggregation considers personalized client information and models each user individually, achieving notable improvements. However, it still incurs some information loss due to item embedding aggregation. In contrast, our Collaborative Information Aggregation effectively aggregates information while preserving the personalized data from each client.

6.4 Model Convergence (RQ2)

In Section 1, we proposed that parameter aggregation leads to more information loss compared to collaborative information aggregation. In this section, we compare the convergence efficiency of them using the Ml-100k and Ml-1m datasets. We do not compare graph-guided aggregation for their method used multiple learning rates with significantly different values, making it difficult to make a fair comparison. The results are illustrated in Figure 3. From the

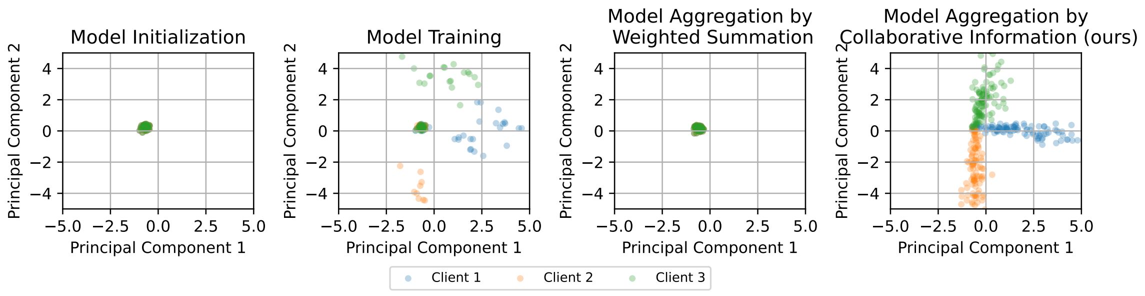

Figure 4: Distribution of item embeddings during different stages of federated learning in different aggregations.

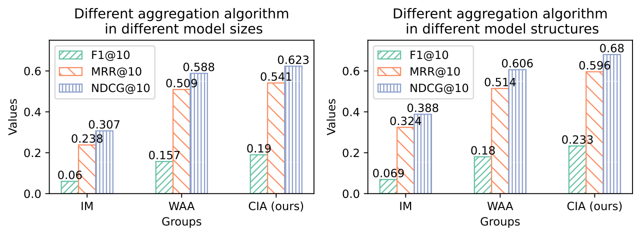

Figure 5: The comparison for different aggregations on heterogeneous scenarios in Ml-100k dataset.

figure, we observe that our approach exhibits better convergence efficiency on both datasets. This demonstrates that the collaborative information aggregation we proposed effectively mitigates the information loss associated with parameter aggregation.

6.5 Personalized Embeddings (RQ2)

We conducted a case study on the Ml-100k dataset to evaluate the effectiveness of personalized item modeling, as illustrated in Figure 1. The federated learning algorithm was divided into three stages:

- Model Initialization: Clients initialize their model with the same distribution.
- Model Training: Clients train their models using their dataset.
- Model Aggregation: Clients aggregate their model using a designated aggregation algorithm.

We randomly selected three clients and 100 items to examine their respective item distributions across these stages. In the third stage, we compared two aggregation paradigms: the weighted summation aggregation and our proposed collaborative information aggregation. The results are depicted in Figure 4.

The figure shows that although each client begins with the same item distribution during initialization, their embeddings diverge significantly after training due to the varied data. The weighted summation aggregation forces all clients into a uniform item distribution, resulting in a loss of personalized information modeling. In contrast, our collaborative information aggregation allows models to integrate collaborative information while maintaining their individual item distributions, effectively leveraging the collaborative data of other items while preserving personalized information, thus achieving better results.

6.6 Scalability Analysis (RQ3)

In this section, we conduct experiments on models in different parameter sizes or different model structures to demonstrate the scalability of our proposed aggregation solution.

In our experiment on model size, we redesigned the federated learning scenario, assigning $\frac{1}{3}$ clients to use a MF model with a 128-dimensional embedding, $\frac{1}{3}$ with a 256-dimensional embedding, and $\frac{1}{3}$ with a 512-dimensional embedding. For the experiment on model structure, we selected equal proportions of MF, MLP, and LightGCN models. Our method is compared with other aggregations, as illustrated in Figure 5.

The independent training model, which cannot aggregate any information, performs poorly. The Weighted Average Aggregation can only aggregate clients with identical model architectures, limiting its ability to gather global collaborative information and leading to performance degradation. In contrast, our method effectively utilizes collaborative information from all clients, regardless of differences in model parameter sizes or structures, resulting in superior performance.

7 Conclusion

In this paper, we propose a novel federated aggregation framework, named FedCIA, for recommendation algorithms. In FedCIA, we utilize collaborative information, specifically item similarity, to perform information aggregation instead of relying on model parameters. Using graph signal processing theory, we demonstrate that the summation of local item similarities can approximate global collaborative information. Experiments on five popular real-world datasets show that FedCIA outperforms existing federated recommendation methods in terms of information aggregation, personalization, and scalability.

Acknowledgments

This work is supported by National Natural Science Foundation of China (NSFC) under the Grant No.62172106. Peng Zhang is a faculty of School of Computer Science, Fudan University. Tun Lu is a faculty of School of Computer Science, Shanghai Key Laboratory of Data Science, Fudan Institute on Aging, MOE Laboratory for National Development and Intelligent Governance, and Shanghai Institute of Intelligent Electronics & Systems, Fudan University.

References

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

[2] Fan RK Chung. 1997. *Spectral graph theory*. Vol. 92. American Mathematical Soc.

[3] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*. 191–198.

[4] EU. 2018. General data protection regulation (GDPR). *Intersoft Consulting, Accessed in October* 24, 1 (2018).

[5] F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)* 5, 4 (2015), 1–19.

[6] Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*. 507–517.

[7] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 639–648.

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

[9] Jonathan L Herlocker, Joseph A Konstan, Al Borchers, and John Riedl. 1999. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*. 230–237.

[10] Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. 2004. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)* 22, 1 (2004), 5–53.

[11] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. 2019. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335* (2019).

[12] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE international conference on data mining*. Ieee, 263–272.

[13] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)* 20, 4 (2002), 422–446.

[14] Hisashi Kashima, Jianying Hu, Bonnie Ray, and Moninder Singh. 2008. K-means clustering of proportional data using L1 distance. In *2008 19th international conference on pattern recognition*. IEEE, 1–4.

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

[16] Mark Levy and Klaas Bosteels. 2010. Music recommendation and the long tail. In *Proceedings of the workshop on music recommendation and discovery (WOMRAD)*. 55–58.

[17] Dongsheng Li, Jianxun Lian, Le Zhang, Kan Ren, Tun Lu, Tao Wu, and Xing Xie. 2024. *Recommender Systems: Frontiers and Practices*. Springer Nature.

[18] Li Li, Yuxi Fan, Mike Tse, and Kuo-Yi Lin. 2020. A review of applications in federated learning. *Computers & Industrial Engineering* 149 (2020), 106854.

[19] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. 2020. Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine* 37, 3 (2020), 50–60.

[20] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems* 2 (2020), 429–450.

[21] Guanyu Lin, Feng Liang, Weike Pan, and Zhong Ming. 2020. Fedrec: Federated recommendation with explicit feedback. *IEEE Intelligent Systems* 36, 5 (2020), 21–30.

[22] Jiahao Liu, Shengkang Gu, Dongsheng Li, Guangping Zhang, Mingzhe Han, Hansu Gu, Peng Zhang, Tun Lu, Li Shang, and Ning Gu. 2025. Enhancing Cross-Domain Recommendations with Memory-Optimized LLM-Based User Agents. *arXiv preprint arXiv:2502.13843* (2025).

[23] Jiahao Liu, Dongsheng Li, Hansu Gu, Tun Lu, Jiongran Wu, Peng Zhang, Li Shang, and Ning Gu. 2023. Recommendation unlearning via matrix correction. *arXiv preprint arXiv:2307.15960* (2023).

[24] Jiahao Liu, Dongsheng Li, Hansu Gu, Tun Lu, Peng Zhang, and Ning Gu. 2022. Parameter-free dynamic graph embedding for link prediction. *Advances in Neural Information Processing Systems* 35 (2022), 27623–27635.

[25] Jiahao Liu, Dongsheng Li, Hansu Gu, Tun Lu, Peng Zhang, Li Shang, and Ning Gu. 2023. Personalized graph signal processing for collaborative filtering. In *Proceedings of the ACM Web Conference 2023*. 1264–1272.

[26] Jiahao Liu, Dongsheng Li, Hansu Gu, Tun Lu, Peng Zhang, Li Shang, and Ning Gu. 2023. Triple structural information modelling for accurate, explainable and interactive recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1086–1095.

[27] Jiahao Liu, Dongsheng Li, Hansu Gu, Peng Zhang, Tun Lu, Li Shang, and Ning Gu. 2025. Mitigating Popularity Bias in Collaborative Filtering through Fair Sampling. *arXiv preprint arXiv:2502.13840* (2025).

[28] Jiahao Liu, Yiyang Shao, Peng Zhang, Dongsheng Li, Hansu Gu, Chao Chen, Longzhi Du, Tun Lu, and Ning Gu. 2024. Filtering Discomforting Recommendations with Large Language Models. *arXiv preprint arXiv:2410.05411* (2024).

[29] Jiahao Liu, Xueshuo Yan, Dongsheng Li, Guangping Zhang, Hansu Gu, Peng Zhang, Tun Lu, Li Shang, and Ning Gu. 2025. Enhancing LLM-Based Recommendations Through Personalized Reasoning. *arXiv preprint arXiv:2502.13845* (2025).

[30] Sijia Liu, Jiahao Liu, Hansu Gu, Dongsheng Li, Tun Lu, Peng Zhang, and Ning Gu. 2023. Autoseqrec: Autoencoder for efficient sequential recommendation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 1493–1502.

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

[32] Vasileios Perifanis and Pavlos S Efraimidis. 2022. Federated neural collaborative filtering. *Knowledge-Based Systems* 242 (2022), 108441.

[33] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*. 285–295.

[34] Yifei Shen, Yongji Wu, Yao Zhang, Caihua Shan, Jun Zhang, B Khaled Letaief, and Dongsheng Li. 2021. How powerful is graph convolution for recommendation?. In *Proceedings of the 30th ACM international conference on information & knowledge management*. 1619–1629.

[35] Daniel A Spielman. 2007. Spectral graph theory and its applications. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*. IEEE, 29–38.

[36] Zehua Sun, Yonghui Xu, Yong Liu, Wei He, Lanju Kong, Fangzhao Wu, Yali Jiang, and Lizhen Cui. 2024. A survey on federated recommendation systems. *IEEE Transactions on Neural Networks and Learning Systems* (2024).

[37] Ben Tan, Bo Liu, Vincent Zheng, and Qiang Yang. 2020. A federated recommender system for online services. In *Proceedings of the 14th ACM conference on recommender systems*. 579–581.

[38] Ellen M Voorhees et al. 1999. The trec-8 question answering track report.. In *Trec*, Vol. 99. 77–82.

[39] Hua Wang, Feiping Nie, and Heng Huang. 2014. Robust distance metric learning via simultaneous l1-norm minimization and maximization. In *International conference on machine learning*. PMLR, 1836–1844.

[40] Jiafeng Xia, Dongsheng Li, Hansu Gu, Jiahao Liu, Tun Lu, and Ning Gu. 2022. FIRE: Fast incremental recommendation with graph signal processing. In *Proceedings of the ACM Web Conference 2022*. 2360–2369.

[41] Jiafeng Xia, Dongsheng Li, Hansu Gu, Tun Lu, Peng Zhang, Li Shang, and Ning Gu. 2024. Hierarchical Graph Signal Processing for Collaborative Filtering. In *Proceedings of the ACM on Web Conference 2024*. 3229–3240.

[42] Jiafeng Xia, Dongsheng Li, Hansu Gu, Tun Lu, Peng Zhang, Li Shang, and Ning Gu. 2024. Neural Kalman Filtering for Robust Temporal Recommendation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. 836–845.

[43] Lianghao Xia, Chao Huang, Yong Xu, Jiashu Zhao, Dawei Yin, and Jimmy Huang. 2022. Hypergraph contrastive collaborative filtering. In *Proceedings of the 45th International ACM SIGIR conference on research and development in information retrieval*. 70–79.

[44] Liu Yang, Ben Tan, Vincent W Zheng, Kai Chen, and Qiang Yang. 2020. Federated recommendation systems. *Federated Learning: Privacy and Incentive* (2020), 225–239.

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

[46] Qiaolin Ye, Henghao Zhao, Zechao Li, Xubing Yang, Shangbing Gao, Tongming Yin, and Ning Ye. 2017. L1-Norm distance minimization-based fast robust twin support vector k-plane clustering. *IEEE transactions on neural networks and learning systems* 29, 9 (2017), 4494–4503.

[47] Chunxu Zhang, Guodong Long, Tianyi Zhou, Peng Yan, Zijian Zhang, Chengqi Zhang, and Bo Yang. 2023. Dual personalization on federated recommendation. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*. 4558–4566.

[48] Chunxu Zhang, Guodong Long, Tianyi Zhou, Zijian Zhang, Peng Yan, and Bo Yang. 2024. GPFedRec: Graph-Guided Personalization for Federated Recommendation. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4131–4142.

[49] Cai-Nicolas Ziegler, Sean M McNee, Joseph A Konstan, and Georg Lausen. 2005. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*. 22–32.