RESEARCH ARTICLE

FedTLRec: Federated Recommendation with Transformer-based Parameter Aggregation and LoRA Compression

Xudong Wang[1,*] **and Ruixin Zhao**[2]

[1] School of Computer Science and Engineering, Tianjin University of Technology, Tianjin 300384, China
[2] Beijing Kunchi Technology Co., Ltd, Beijing, China

Abstract

Federated learning has emerged as a key paradigm in privacy-preserving computing due to its "data usable but not visible" property, enabling users to collaboratively train models without sharing raw data. Motivated by this, federated recommendation systems offer a promising architecture that balances user privacy with recommendation accuracy through distributed collaborative learning. However, existing federated recommendation systems face significant challenges in balancing model performance, communication efficiency, and user privacy. In this paper, we propose FedTLRec (Federated Recommendation with Transformer-based Parameter Aggregation and Collaborative LoRA), which introduces a federated recommendation framework that integrates Low-Rank Adaptation (LoRA) for parameter compression and Transformer-based aggregation. It addresses key challenges in communication efficiency and model performance by compressing client updates via LoRA and employing a Transformer model with attention mechanisms to effectively aggregate parameters from multiple clients. A K-means clustering strategy further enhances efficiency by grouping similar clients. Experiments on real-world datasets show that FedTLRec achieves superior recommendation accuracy with significantly reduced communication costs, while maintaining robust performance in client dropout scenarios. Code is available at: https://github.com/trueWangSyutung/FedTLRec.

Keywords: federated recommendation, low-rank adaptation, transformer.

1 Introduction

The proliferation of digital services has intensified the demand for privacy-preserving recommendation systems that can provide personalized experiences without compromising user data confidentiality. Traditional centralized recommendation systems require collecting vast amounts of sensitive user data in a central repository, raising serious privacy concerns and legal compliance issues. Federated learning has emerged as a promising paradigm that enables collaborative model training without direct data sharing, offering a viable solution to this challenge [1–3].

Submitted: 30 December 2025
Accepted: 20 January 2026
Published: 08 February 2026

Vol. 2, **No.** 1, 2026.
10.62762/TMI.2025.882476

***Corresponding author:**
✉ Xudong Wang
wangxudong@stud.tjut.edu.cn

Citation
Wang, X., & Zhao, R. (2026). FedTLRec: Federated Recommendation with Transformer-based Parameter Aggregation and LoRA Compression. *ICCK Transactions on Machine Intelligence*, 2(1), 65–76.

The traditional recommendation system paradigm in federated learning (such as FedMF [4], FedNCF [5]) treats project embedding as a global model and user embedding as a local variable. This approach is not optimal due to natural heterogeneity between clients. For example, clients often have non-independent and identically distributed (Non-IID) interactive data [1, 6]. As a result, personalized federated recommendations have been developed to create customized models for each customer [7, 8].

The main challenges of current federated recommendation systems include: 1) **Communication Overhead**: Traditional federated learning methods require transmitting complete model parameters between the client and server. This is especially problematic for large-scale recommendation models with extensive embedding layers. 2) **Personalization and Collaboration**: These systems must balance global knowledge sharing with individual user preferences. They also need mechanisms that use collective intelligence while providing personalized recommendations. 3)**Parameter Aggregation Efficiency**: Simple averaging of client parameters may not capture complex relationships and similarities between different clients.

To address these challenges, we propose FedTLRec, a novel federated recommendation framework that combines LoRA (Low-Rank Adaptation) parameter compression with Transformer-based parameter aggregation. Our approach is motivated by the observation that client-side model updates often reside in a low-dimensional subspace, making them amenable to compression techniques. Additionally, we recognize that effective parameter aggregation should consider the relationships between clients rather than simply averaging their parameters.

The key contributions of our work are as follows:

- We introduce a LoRA-based parameter compression technique that significantly reduces communication overhead in federated recommendation systems.
- We introduce a Transformer-based parameter aggregation model that effectively captures client relationships and similarities, leading to improved recommendation performance.
- We implement a K-means clustering strategy to group clients with similar characteristics before parameter aggregation, further enhancing efficiency and performance.
- We conduct extensive experiments on four real-world datasets, demonstrating that FedTLRec outperforms state-of-the-art federated recommendation methods in both recommendation accuracy and communication efficiency.

2 Related Work

2.1 Federated Recommendation Systems

Federated Recommendation System (FedRS) is an important extension of Federated Learning (FL) in the field of recommendation, which achieves cross client personalized modeling by protecting user privacy. Early works such as FedMF [4] and FedNCF [5] adapted classic recommendation models to federated settings by treating project embeddings as global parameters and user embeddings as local parameters. These methods successfully protect user privacy while maintaining reasonable recommendation performance.

Recent research has mainly focused on improving personalization in federated recommendation systems. PFedGraph [9] models user relationships through graph construction and performs personalized aggregation based on user similarity. GPFedRec [10] introduces a graph-guided aggregation mechanism to learn user-specific item embeddings. PFedRec [7] adopts a dual personalization mechanism that captures user preferences through a scoring function while optimizing global item embeddings. FedRAP [11] balances global shared knowledge and local personalized knowledge through an additive model. UFGraphFR [12] utilizes user text feature descriptions and transformer modeling of interactive items to enhance the performance of recommendation systems.

Although these methods have shown good results in terms of effectiveness, they often suffer from high communication costs due to insufficient consideration of client relationships in transmitting complete model parameters and suboptimal parameter aggregation.

2.2 Parameter-Efficient Fine-Tuning

In recent years, with the development of natural language processing technology and computer vision technology, parameter efficient fine-tuning techniques have become increasingly popular. This is a method of adapting large pre trained models with minimal additional parameters. LoRA [13] (low rank adaptive) is a technique that decomposes weight updates into low rank matrices, significantly reducing the number

of trainable parameters. AdaLoRA [14] extends LoRA by dynamically adjusting the rank of decomposition based on parameter importance. These technologies show great potential in reducing model size and training time while maintaining performance.

In the context of federated learning, FedLoRA [21] applies LoRA to reduce communication costs in federated recommendation systems. PFedCLR [15] brings LoRA technology into the field of federated recommendation systems, introducing an integrated dual function mechanism implemented through a buffer matrix to perform federated calibration of local user embeddings and personalized global item embeddings. However, in the above methods, they all adopt a simple parameter averaging aggregation approach without considering more complex parameter aggregation mechanisms.

2.3 Transformer-Based Models

Transformers [16] have revolutionized many areas of machine learning through their ability to capture long-range dependencies and complex relationships. In recommendation systems, SASRec [17] and BERT4Rec [18] have demonstrated the effectiveness of self-attention mechanisms for sequential recommendation. In federated learning, recent works have explored using attention mechanisms for client relationship modeling. FedFormer [19] employs transformers for personalized federated learning, while FedTrans [20] uses transformer-based attention for client selection. However, these approaches focus on general federated learning scenarios rather than recommendation-specific challenges. However, these approaches focus on general federated learning scenarios rather than recommendation-specific challenges.

In our previous research on UFGraphFR [12], we placed the converter on the client for sequence modeling, which improved model performance but significantly reduced local computation speed.

Our work combines these three lines of research by applying LoRA for parameter compression, Transformer-based models for parameter aggregation, and federated learning principles for privacy-preserving recommendations.

3 Methodology

The U is user sets and the I is item sets, respectively, let r_{ui} be user-item interaction data between user u and item i. Here is a recommendation system model f for parameter θ, which predicts $\hat{y_{ui}} = f(u, i|\theta)$ for users u and i. On the central server, we represent the graph between all users with $\mathcal{G}(\mathcal{U}, \mathcal{E})$, where $\mathcal{U}$ represents the set of users and $\mathcal{E}$ represents the set of edges. At the same time, $\mathcal{A} = \{0, 1\}^{N \times N}$ represents its corresponding adjacency matrix form, and N represents the total number of users in the user set. $\mathcal{A}_u i = 1$ indicates that user u and user i are associated.

For the above model, the goal of the federated recommendation system is to predict user u's preference for item i as $\hat{y_{ui}} = f(u, i|\theta^*)$. At this point, for each user i, there is an optimal model parameter θ_i^*, as shown in formula (1).

$$\theta_i^* = argmin_\theta = \sum_{i=1}^{N} \omega_i \mathcal{L}_i(\theta) \tag{1}$$

where $\mathcal{L}_i(\theta)$ is the loss of the local client participating in training, and the parameter θ^* is learned by minimizing the local loss of all clients with weight ω_i.

In our method, the optimal parameter θ_i^* for the user includes the local training parameter θ_i^{local} and the global parameter θ_i^{global}. The global parameter θ_i^{global} is aggregated through the Transfermer module.

4 FedTLRec: Proposed Method

In this section, we present FedTLRec, our novel federated recommendation framework. As illustrated in Figure 1, our approach consists of three key components: 1) a LoRA-based parameter compression module that reduces communication overhead; 2) a K-means clustering strategy that groups similar clients for more efficient aggregation; and 3) a Transformer-based parameter aggregation model that effectively integrates client updates.

4.1 LoRA-based Parameter Compression

In federated recommendation systems, a significant portion of communication overhead stems from transmitting large embedding matrices between clients and the server. To address this challenge, we employ LoRA (Low-Rank Adaptation) [13] to compress client model updates.

Specifically, for each client u, we decompose the item embedding matrix $\mathbf{E}_u \in \mathbb{R}^{|\mathcal{I}| \times d}$ (where $|\mathcal{I}|$ is the number of items and d is the embedding dimension) into a low-rank form:

$$\mathbf{E}_u^t = \mathbf{E}_u^{t-1} + \Delta \mathbf{E}_u \tag{2}$$

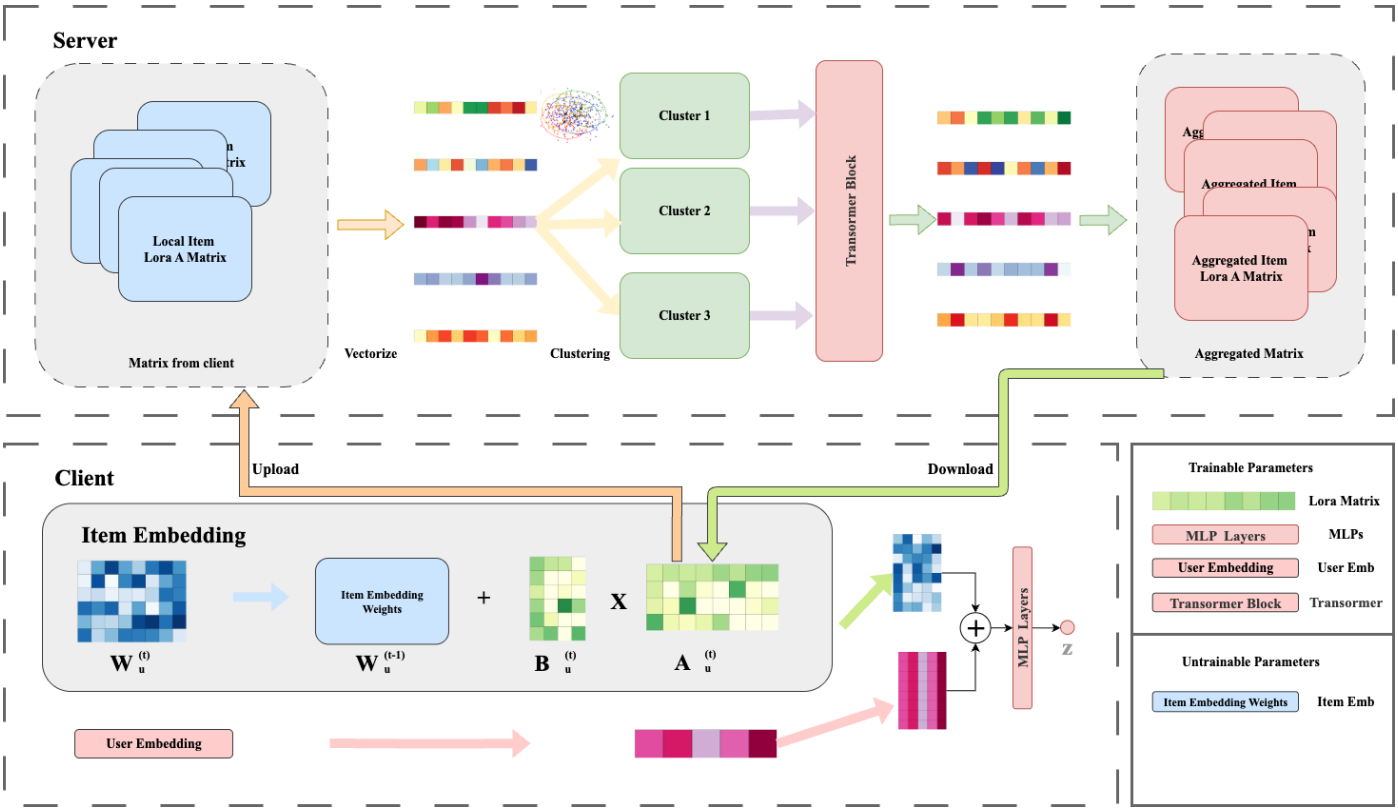

Figure 1. Overview of FedTLRec. During local training, clients first train the user embeddings, parameters of the MLP, and the LoRA matrices A and B locally. At the same time, clients upload their LoRA matrix A to the server, which then aggregates these matrices. The server only maintains a clustering module and a Transformer module, which it uses to aggregate the uploaded user-related parameters.

where $\mathbf{E}_u^{t-1}$ is the item embedding matrix for user u at the previous communication round $t-1$, and the low-rank update term $\Delta\mathbf{E}_u$ is parameterized as:

$$\Delta\mathbf{E}_u = \alpha\,\mathbf{B}_u\mathbf{A}_u \tag{3}$$

with the following mathematical specifications:

- $\mathbf{E}_u \in \mathbb{R}^{|\mathcal{I}|\times d}$: the full item embedding matrix for client u;
- $\mathbf{A}_u \in \mathbb{R}^{r\times d}$: trainable low-rank adaptation matrix (right factor);
- $\mathbf{B}_u \in \mathbb{R}^{|\mathcal{I}|\times r}$: trainable low-rank projection matrix (left factor);
- $r \in \mathbb{N}^+$: the rank of the low-rank update, satisfying $r \ll \min(|\mathcal{I}|, d)$;
- $\alpha \in \mathbb{R}$: scalar scaling factor, often fixed as $\alpha = \frac{\beta}{r}$ with β a tunable hyperparameter.

In our experimental setting, we adopt the constraint $r = d$, which simplifies the parameterization while still enabling substantial compression when $d \ll |\mathcal{I}|$. Under this condition, the low-rank update $\Delta\mathbf{E}_u = \alpha\,\mathbf{B}_u\mathbf{A}_u$ remains expressive yet compact.

During federated training, instead of transmitting the full embedding matrix $\mathbf{E}_u \in \mathbb{R}^{|\mathcal{I}|\times d}$, each client uploads only the smaller matrix $\mathbf{A}_u \in \mathbb{R}^{r\times d}$, assuming $\mathbf{B}_u$ is either fixed (e.g., initialized from the global item embedding and frozen) or reconstructable on the server using shared context. The resulting communication cost per client is reduced from $|\mathcal{I}| \cdot d$ to $r \cdot d$ floating-point values. Hence, the compression rate ρ defined as the ratio of transmitted parameters to the original embedding size is:

$$\rho = \frac{r \cdot d}{|\mathcal{I}| \cdot d} = \frac{r}{|\mathcal{I}|} \tag{4}$$

Notably, the embedding dimension d cancels out, and the compression efficiency depends solely on the relative scale of the rank r to the item catalog size $|\mathcal{I}|$.

In practice, with the typical values used in our experiments $r = d = 16$ and $|\mathcal{I}| = 1000$ the compression ratio becomes:

$$\rho = \frac{16}{1000} = 0.016 = 1.6\% \tag{5}$$

This corresponds to a communication overhead reduction factor of: $1/\rho = 1000/16 = 62.5$. Thus, our

LoRA-based update scheme achieves approximately a **62.5**× reduction in uplink communication cost.

4.2 Transformer-based Parameter Aggregation

To effectively aggregate heterogeneous client parameter updates in federated recommendation systems, we propose a **structured Transformer-based aggregation mechanism** that explicitly models inter-client dependencies. Unlike conventional averaging-based strategies (e.g., FedAvg), our approach leverages self-attention to dynamically weigh each client's contribution based on the representational similarity of their local parameters.

Let $\mathcal{C} = \{c_1, c_2, \ldots, c_n\}$ denote the set of n participating clients. Each client c_i uploads its local LoRA adapter parameters $\theta_i \in \mathbb{R}^{r \times d}$, where r is the LoRA rank and d is the embedding dimension. We first **flatten and project** each θ_i into a token representation:

$$x_i = \mathrm{Proj}(\mathrm{vec}(\theta_i)) \in \mathbb{R}^h,$$

where $\mathrm{vec}(\cdot)$ vectorizes the matrix, and $\mathrm{Proj} : \mathbb{R}^{rd} \to \mathbb{R}^h$ is a trainable linear projection shared across clients. The server thus receives an input sequence $X = [x_1, x_2, \ldots, x_n]^\top \in \mathbb{R}^{n \times h}$.

Direct application of a standard Transformer on X incurs $\mathcal{O}(n^2 h)$ self-attention complexity, which becomes prohibitive when $n \gg 10^3$. To mitigate this, we adopt **block-wise attention**: partition X into k non-overlapping blocks $\{X^{(1)}, \ldots, X^{(k)}\}$, each of length $m = \lceil n/k \rceil$. Within each block, we apply an independent Transformer encoder:

$$\tilde{X}^{(j)} = \mathrm{TransformerBlock}(X^{(j)}), \quad j = 1, \ldots, k,$$

where the Transformer block consists of **multi-head self-attention (MHSA)** followed by a **position-wise feed-forward network (FFN)**.

The MHSA is formally defined as:

$$\mathrm{MHSA}(X) = \mathrm{Concat}(\mathrm{head}_1, \ldots, \mathrm{head}_H) W^O, \quad (6)$$

with each head computed as:

$$\mathrm{head}_i = \mathrm{softmax}\left(\frac{(XW_i^Q)(XW_i^K)^\top}{\sqrt{d_k}}\right)(XW_i^V), \quad (7)$$

where $W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{h \times d_k}$ are learnable projection matrices for the i-th head, $d_k = h/H$, and $W^O \in \mathbb{R}^{h \times h}$ is an output projection.

The FFN is defined as:

$$\mathrm{FFN}(Z) = \mathrm{GeLU}(ZW_1 + b_1)W_2 + b_2, \quad (8)$$

with $W_1 \in \mathbb{R}^{h \times d_{\text{ff}}}$, $W_2 \in \mathbb{R}^{d_{\text{ff}} \times h}$, and d_{ff} typically set to $4h$.

After processing all blocks, we **concatenate** the outputs and apply a **global aggregation layer** to produce the final aggregated parameter tokens:

$$\bar{X} = \mathrm{Concat}\left(\tilde{X}^{(1)}, \ldots, \tilde{X}^{(k)}\right) \in \mathbb{R}^{n \times h}, \quad (9)$$

To effectively aggregate heterogeneous client parameter updates and generate personalized model parameters, we define the aggregation operation as follows. For each token $\bar{x}_i$ in the processed sequence $\bar{X} = [\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n]^T \in \mathbb{R}^{n \times h}$, a personalized parameter update is generated:

$$\theta_i^{\text{agg}} = \mathrm{Reshape}(\bar{x}_i) \quad \text{for } i = 1, \ldots, n, \quad (10)$$

where $\mathrm{Reshape}(\cdot)$ is to reshape the vector into a matrix form and send the result to the client

Complexity Analysis. The block-wise design reduces the attention complexity from $\mathcal{O}(n^2 h)$ to $\mathcal{O}(km^2 h) = \mathcal{O}\left(\frac{n^2 h}{k}\right)$. By choosing $k \propto n$ (e.g., fixed block size $m = 512$), the complexity becomes **linear in** n, i.e., $\mathcal{O}(nh)$, enabling scalability to thousands of clients.

This Transformer-based approach allows the server to effectively capture relationships between different clients, leading to more informed parameter aggregation compared to simple averaging methods.

In our implementation, we split the client parameters into multiple groups based on the maximum line parameter to handle large numbers of clients efficiently. For each group, we train a separate Transformer block to process the parameters and then combine the outputs to form the final aggregated parameters.

Algorithm 1 FedTLRec: Federated Recommendation with Transformer-based Parameter Aggregation and LoRA Compression

Require: Number of clients N, total communication rounds T, local epochs E, learning rate η, regularization coefficient λ, LoRA parameters r, α, and other local model parameters θ_{local}.
Ensure: Optimized global item embedding θ_{global}.

Initialize global parameters $\theta_{global}^{(0)}$ (e.g., item embeddings) on the server.
Broadcast $\theta_{global}^{(0)}$ to all N clients.
for each communication round $t = 1, 2, ..., T$ **do**
  Client Selection: Randomly select $c \times N$ clients to participate in this round, where c is the client sampling ratio.
  Client Local Training (Parallel Execution):
  for each selected client i **do**
    Initialize local item embedding $\theta_{item,i}^{(t)} \leftarrow \theta_{global}^{(t-1)}$.
    Use local data to train the local model for E epochs by minimizing the loss function $\mathcal{L}_{all} = \mathcal{L}_{BCE} + \lambda \cdot \mathcal{L}_{reg}$ via SGD.
    Obtain updated local LoRA parameters: $\theta_{lora,i}^{(t)}$.
    Offline Simulation: With probability o, set client i as offline and upload zero parameters.
    Privacy Protection: Add Laplace noise to parameters with scale Δ for differential privacy.
    Upload the local LoRA parameters $\theta_{lora,i}^{(t)}$ to the server.
  end for
  Server Update with Graph Aggregation:
  Vectorize: Flatten each client's uploaded LoRA parameters: $w_i^{(t)} = \text{vec}(\theta_{lora,i}^{(t)}) \in \mathbb{R}^{r \times d}$.
  Clustering (Optional): Apply K-means clustering to group users based on parameter similarity.
  Transformer Aggregation: Apply Transformer-based aggregation model to local parameters within each group.
  Update Client Embeddings: Generate personalized parameters $\theta_i^{(t+1)}$ for each client i.
  Broadcast $\theta_i^{(t+1)}$ to client i.
end for
Evaluation: Evaluate the final model on test data using Hit Ratio and NDCG metrics.
return The final global item embedding $\theta_{global}^{(T)}$.

As detailed in Algorithm 1, this iterative process continues for T communication rounds, progressively optimizing the global item embeddings. Finally, the trained model is evaluated using standard recommendation metrics such as Hit Ratio and NDCG, yielding a scalable, robust, and personalized federated recommendation system.

4.3 K-means Clustering for Efficient Aggregation

To further enhance the efficiency of parameter aggregation, particularly in scenarios involving a large number of clients, we employ a K-means clustering strategy. This approach groups clients with similar characteristics together, allowing for aggregation of more targeted parameters within each group.

The clustering process works as follows:

1. For each client u, we vectorize heir LoRA parameters $\mathbf{A}_u$ to $\mathbf{f}_u$.
2. We apply K-means clustering to group clients into K clusters: $\mathcal{C} = \{C_1, C_2, \ldots, C_K\}$.
3. Within each cluster C_k, we apply the Transformer-based aggregation model independently.
4. Finally, we combine the cluster-level aggregated parameters to form the global model update.

In our implementation, we perform clustering every 10 rounds to reduce computational overhead, as client similarities may change gradually over time.

4.4 Handling Client Dropout

In practical federated learning scenarios, some clients may be offline or fail to participate in certain rounds. Our framework naturally handles client dropout by simply excluding the missing clients from the current round's aggregation process. Additionally, we implement a zero-padding strategy for offline clients, where their parameters are set to zero during aggregation, ensuring that the server can still produce meaningful aggregated parameters even with partial client participation.

4.5 Algorithm

We propose FedTLRec, a federated transfer learning framework for recommendation that enables efficient and privacy-preserving collaborative model training across decentralized clients. By leveraging low-rank adaptation (LoRA), each client performs local fine-tuning using its private interaction data, updating only a small set of low-dimensional parameters,

thereby reducing communication overhead. After local updates, clients optionally inject Laplace noise for differential privacy and may be marked as offline with probability o to simulate real-world deployment scenarios. The server aggregates the uploaded LoRA parameters by first vectorizing them and, optionally, clustering clients based on parameter similarity to capture behavioral heterogeneity. A Transformer-based aggregation module is then applied within each cluster to model client relationships and generate personalized global models through attention-driven fusion. These personalized models are broadcast back to the respective clients for the next round. As detailed in Algorithm 1, this iterative process continues for T communication rounds, progressively optimizing the global item embeddings. Finally, the trained model is evaluated using standard recommendation metrics such as Hit Ratio and NDCG, yielding a scalable, robust, and personalized federated recommendation system.

5 Experiment

5.1 Datasets

To assess the performance of our proposed model, we performed comprehensive experiments on four well-established benchmark datasets in the recommendation domain: MovieLens-100K, MovieLens-1M [22], Lastfm-2K [23], and HetRec2011 [23]. The MovieLens-100K and MovieLens-1M datasets, sourced from the MovieLens platform, contain user-generated movie ratings with each user having provided at least 20 ratings. Lastfm-2K documents music listening behaviors, including artists played and frequency of plays per user. We refined this dataset by filtering out users with fewer than 5 interactions. HetRec2011 extends MovieLens-10M by incorporating mappings to IMDb and Rotten Tomatoes pages for enhanced movie metadata. For Lastfm-2K and HetRec2011, user profiles were derived from user identifiers and their total interaction counts with items. Table 1 summarizes the essential statistics of these datasets.

Table 1. Dataset statistics.

Dataset	Users	Items
ml-1m	6,040	3,706
lastfm-2k	1,600	12,454
hetres-2k	2,113	10,109
100k	943	1,682

5.2 Baselines

Our proposed method is evaluated against two primary categories of baseline approaches: centralized recommendation models and federated recommendation models. All methods compared rely solely on user-item interaction data to generate recommendations, ensuring a fair comparison.

- **Matrix Factorization (MF)** [24]: A foundational collaborative filtering technique that decomposes the user-item interaction matrix into lower-dimensional latent factor representations. These factors capture underlying user preferences and item characteristics within a shared latent space, enabling the prediction of unknown interactions through dot products of the learned embeddings.
- **Neural Collaborative Filtering (NCF)** [25]: An influential neural network-based recommendation model that employs embedding layers to represent users and items separately. It utilizes a multi-layer perceptron (MLP) to model complex, non-linear interactions between these embeddings, predicting user preference scores effectively.
- **LightGCN** [26]: A streamlined graph convolutional network designed specifically for collaborative filtering. It simplifies traditional GCN architectures by focusing solely on neighborhood aggregation, which efficiently propagates collaborative signals on the user-item interaction graph to learn enhanced representations.
- **FedMF** [4]: A federated adaptation of matrix factorization where clients perform local updates to user embeddings. Only gradient updates related to item embeddings are transmitted to the server for secure global aggregation, thereby maintaining user data privacy throughout the process.
- **FedNCF** [5]: A federated variant of the Neural Collaborative Filtering model. In this framework, user embeddings remain as private parameters on local devices, while item embeddings and the MLP interaction module are trained collaboratively across clients via federated averaging on the server.
- **pFedGraph** [9]: A personalized federated recommendation approach that integrates graph neural networks to capture local user-item

interaction patterns. It employs a personalized aggregation strategy to preserve and leverage client-specific behavioral patterns during the federated training process.

- **Graph-Guided Personalization for Federated Recommendation (GPFedRec)** [10]: A federated recommendation system that utilizes a graph-based aggregation mechanism informed by client-level similarity relationships. This method explicitly models correlations between users to enhance recommendation performance and has significantly influenced our design.
- **Personalized Federated Recommendation (PFedRec)** [7]: framework designed for personalized federated learning where a global item embedding is maintained on the server. Each client then performs localized fine-tuning of this global embedding to better align with its unique data distribution and user preferences.
- **FedRAP** [11]: A federated recommendation approach featuring adaptive personalization, which employs meta-learning principles to dynamically balance the contributions of global model updates and local adaptations throughout the training process.
- **PFedCLR** [11]: A federated recommendation approach leverages LoRA and introduces a dual-function mechanism implemented via a buffer matrix to jointly calibrate local user embeddings and personalized global item embeddings.

5.3 Performance Comparison

Table 2 shows the performance comparison between FedTLRec and baseline methods across all datasets. Our method outperforms most baselines in two aspects HR@10 and NDCG@10 This proves the effectiveness of our method.

We implemented this framework based on PyTorch. To ensure a fair comparison, we set the embedding layer dimension to 16, the client MLP layer dimensions to 32→16→8→1, the LoRA parameter $r = 16$, the LoRA scaling factor $lora\_alpha = 16$, a batch size of 128, a learning rate of 0.001, 1 local training epoch, and 100 global rounds to allow for sufficient convergence. In each round, all client machines participated in the training (though 20% of the clients were offline). On the server side, a K-means clustering mechanism was first employed to partition the clients into 4 clusters every 10 rounds. Subsequently, each cluster was fed into a Transformer model for parameter aggregation. During the Transformer computation, the parameters uploaded by users were processed in groups of 128 clients (padding with zeros when insufficient).

From the results, we can make several key observations:

1. Our method (FedTLRec) achieves the best or second-best performance across all four datasets (ML-100K, ML-1M, HetRec2011, LastFM-2K) in both HR@10 and NDCG@10. Notably, it consistently outperforms all existing FedRS methods when LDP is applied. Compared to the strongest non-private FedRS baseline (PFedCLR), FedTLRec improves NDCG@10 by up to **13.12%** on ML-1M and **8.42%** on HetRec2011, while maintaining competitive HR@10 (e.g., only 0.19% lower than PFedCLR on ML-100K).
2. On the ML-1M dataset, FedTLRec achieves a near-perfect HR@10 of **0.9998** and an NDCG@10 of **0.9505**, representing a **13.12%** relative improvement in NDCG@10 over PFedCLR the previous best method. Similarly, on HetRec2011, it obtains an NDCG@10 of **0.9212**, surpassing PFedCLR by **8.42%**, which highlights its superior ability to produce accurate and well-ranked recommendations under data heterogeneity.
3. Despite applying local differential privacy (LDP), FedTLRec demonstrates robust performance on the LastFM-2K dataset, which contains a large and sparse item space (over 12,000 music artists). It achieves the highest HR@10 (**0.7919**) and NDCG@10 (**0.7229**) among all methods, improving upon the best baseline (GPFedRec in HR@10, PFedCLR in NDCG@10) by **0.2%** and **0.9%**, respectively. This confirms that our framework maintains effectiveness even in complex, large-scale settings while preserving user privacy.

5.4 Ablation Study

5.4.1 Lora's rank

To validate the effectiveness of LoRA (Low-Rank Adaptation) in FedTLRec and explore the impact of its key hyperparameter, the adaptation rank (r), on model convergence and recommendation performance, we designed a series of ablation experiments. All experiments were conducted on the MovieLens-100k dataset, using the same federated training framework (FedTLRec). We fixed the client sampling ratio

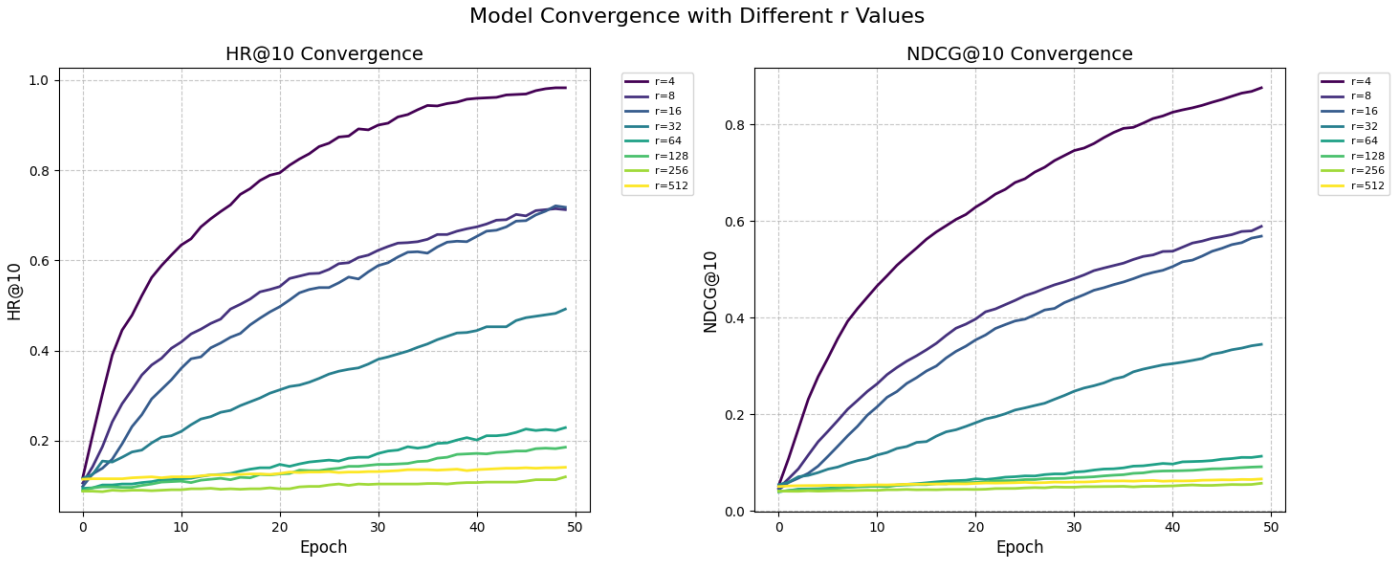

Figure 2. Trend of the hit rate (HR@10) of the model on the test set with different r values as the training rounds change.

to 100%, the number of local training rounds to 1, and the total number of communication rounds to 50. We only adjusted the LoRA rank $r \in \{4, 8, 16, 32, 64, 128, 256, 512\}$.

It's worth noting that if LoRA is not implemented at all (i.e., full parameter fine-tuning is employed), the server-side aggregation module, based on the Transformer architecture, must receive and fuse the complete model parameters uploaded by all clients, leading to a dramatic increase in server-side memory and computational overhead. In this experimental setup (943 users, 1682 items, 16 potential dimensions), the full-parameter version of the model would require over 10 million parameters, making efficient aggregation and training impossible on conventional hardware. Therefore, LoRA is not only an efficient parameter fine-tuning strategy but also a key technical prerequisite for achieving the scalability of this framework.

Figure 2 (with a curve) shows the model's hit rate (HR@10) on the test set as a function of training epochs for different r values. Experimental results show that the rank of LoRA has a significant nonlinear effect on the model's convergence speed and final performance:

Table 2. Performance comparison on four datasets. The best FedRS results are **bolded** and the second best are underlined. **Ours** (**FedTLRec**) applies local differential privacy (LDP). "Improvement" is calculated over the best existing FedRS method.

Method	ML-100K		ML-1M		HetRec2011		LastFM-2K	
	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10	HR@10	NDCG@10
MF	0.6543	0.3788	0.6088	0.3446	0.6275	0.3688	0.8440	0.6191
NCF	0.6119	0.3422	0.5858	0.3267	0.6171	0.3663	0.7896	0.6069
LightGCN	0.6787	0.3994	0.6684	0.3885	0.6611	0.3975	0.8448	0.6853
FedMF	0.4846	0.2723	0.4876	0.2734	0.5376	0.3206	0.5839	0.3930
FedNCF	0.4252	0.2290	0.4180	0.2311	0.5083	0.2982	0.4933	0.3220
pFedGraph	0.6204	0.4937	0.7262	0.5991	0.6962	0.5523	0.6485	0.6085
GPFedRec	0.6840	0.3982	0.6836	0.4012	0.6488	0.4016	0.7896	0.6499
PFedRec	0.6702	0.3929	0.6611	0.3849	0.6531	0.3948	0.7549	0.6634
FedRAP	0.8823	0.7980	0.8661	0.7666	0.8486	0.6325	0.6257	0.5924
PFedCLR	0.9989	0.9225	0.9603	0.8402	0.9522	0.8496	0.7778	0.7164
FedTLRec	0.9979	0.8405	**0.9998**	**0.9505**	**0.9749**	**0.9212**	**0.7919**	**0.7229**
Improvement	↓ **0.19%**	↓ **8.8%**	↑ **4.1%**	↑ **13.12%**	↑ **2.3%**	↑ **8.42%**	↑ **0.2%**	↑ **0.9%**

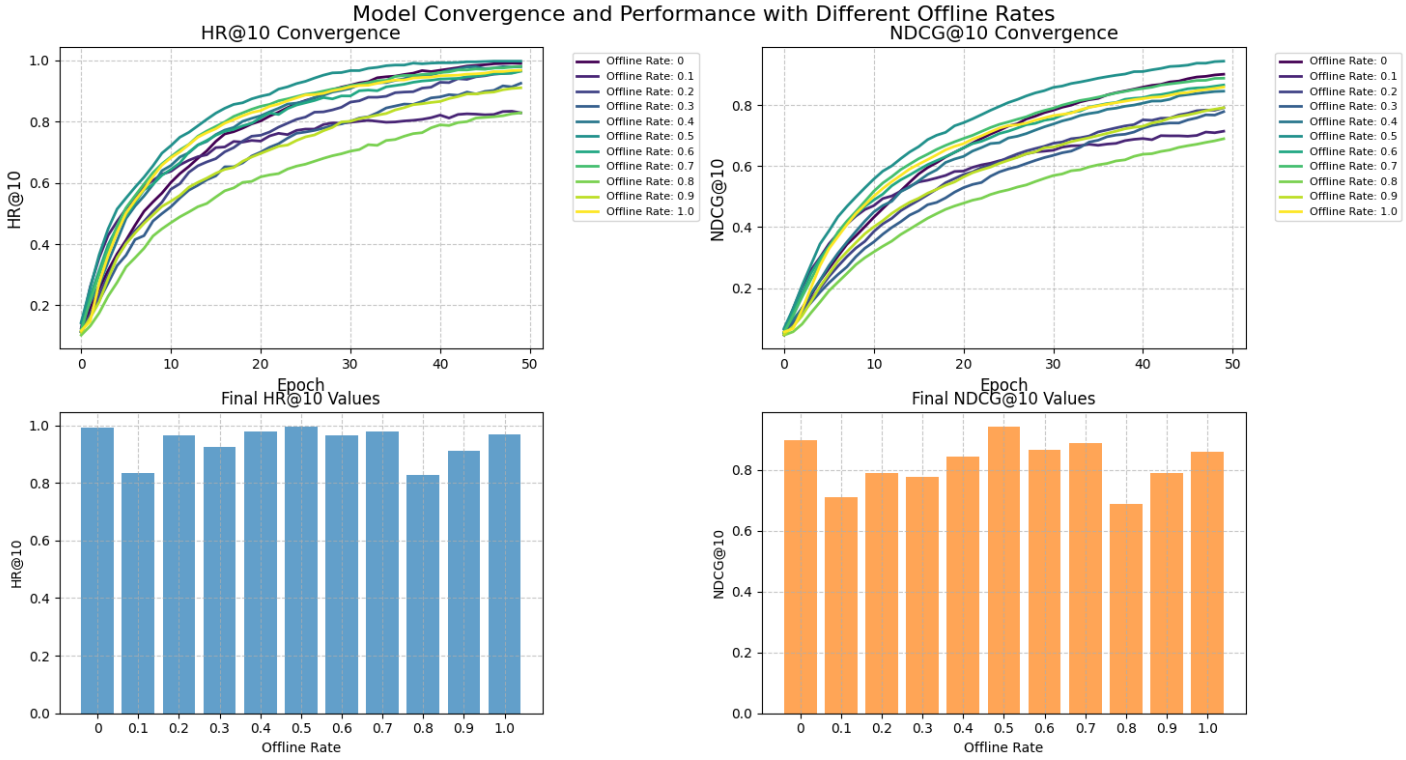

Figure 3. Trend of the hit rate (HR@10) of the model on the different offline rate.

When $r = 4$, the model exhibits the best overall performance: it converges around epoch 30, reaching a final test HR of 0.9830 and a validation HR of 0.9894, significantly outperforming other configurations. When $r = 8$ or $r = 16$, the model still converges stably, but the final HRs are 0.7147 and 0.7179, respectively, indicating a significant performance decline. When $r \geq 32$, the model's convergence speed slows sharply, and final performance continues to deteriorate. For example, when r=64, the final HR is only 0.2227, while when $r = 512$, it drops to 0.1410. The training curve does not saturate within 50 epochs. This phenomenon suggests that in the federated graph recommendation scenario, an excessively large LoRA rank can actually harm the model optimization process. We speculate that the reasons for this are: (1) high rank introduces too many degrees of freedom, making effective learning difficult in a federated setting with sparse data (each client holds only a portion of the interaction records) and limited communication (only one aggregation per round); and (2) the regularization term (reg = 0.1) and dropout (dp = 0.1) increase the penalty effect on large-capacity adapters, inhibiting the transmission of effective signals.

5.4.2 *Offline rate*

In real-world federated recommendation scenarios, client devices (such as mobile terminals or IoT devices) often experience intermittent offline states due to network fluctuations, battery constraints, or user behavior. To evaluate the robustness of FedTLRec under such non-ideal communication conditions, we systematically investigate the impact of client offline rate on model performance.

Specifically, we fixed the LoRA rank r = 4 on the MovieLens-100k dataset (as shown in Section 4.3, this is the optimal configuration) and only adjusted the offline rate $p_{off} \in 0.0, 0.1, 0.2, \ldots, 1.0$ for 11 sets of experiments. Here, $p_{off} = 0$ indicates that all clients are always online, while $p_{off} = 1$ indicates that all clients are marked as "offline" with probability 1 (i.e., do not upload parameters) in each training round. However, because the client sampling rate is set to 100%, they still participate in local training and parameter upload (simulating an extremely unstable connection).

Figure 3 demonstrates FedTLRec's robust performance under varying client dropout conditions. The model achieves peak performance with moderate dropout ($p_{\text{off}} = 0.5$, HR@10=0.9968), where client absence acts as beneficial regularization. It also maintains strong stability under high dropout ($p_{\text{off}} \geq 0.8$, HR@10>0.82), confirming its resilience in unstable environments.

Performance dips at both low ($p_{\text{off}} = 0.1$) and high ($p_{\text{off}} = 0.8$) dropout rates (HR@10=0.8282).

For low dropout, sporadic client absence disrupts aggregation consistency without activating robustness mechanisms; for high dropout, sparse effective updates hinder information fusion and optimization. These results collectively validate FedTLRec's adaptability across diverse connectivity scenarios.

6 Conclusion

In this paper, we presented FedTLRec, a novel federated recommendation framework that combines LoRA-based parameter compression with Transformer-based parameter aggregation. Our approach addresses key challenges in federated recommendation systems, including communication overhead, parameter aggregation efficiency, and model personalization.

The main contributions of our work are threefold: 1) We use LoRA to significantly reduce communication overhead by compressing client model updates, reducing data transmission by several times; 2) We introduce a Transformer-based parameter aggregation model that effectively captures relationships between clients, leading to improved recommendation performance; 3) We implement a K-means clustering strategy to further enhance aggregation efficiency and performance.

Extensive experiments on four real-world datasets demonstrate that FedTLRec consistently outperforms state-of-the-art federated recommendation methods in terms of both recommendation accuracy and communication efficiency. Our ablation studies confirm the effectiveness of each component in our framework, and additional experiments show that FedTLRec is robust to client dropout and can maintain high performance even in challenging federated learning environments. In our implementation, we handle large numbers of clients efficiently by splitting them into groups based on a maximum line parameter, and we perform clustering every 10 rounds to reduce computational overhead.

For future work, we plan to explore more sophisticated client clustering methods that can dynamically adapt to changing client characteristics. Additionally, we aim to investigate the application of our framework to other federated learning scenarios beyond recommendation systems.

Data Availability Statement

The source code supporting the findings of this study is publicly available at: https://github.com/trueWangSyutung/FedTLRec.

Funding

This work was supported without any funding.

Conflicts of Interest

Ruixin Zhao is affiliated with the Beijing Kunchi Technology Co., Ltd, Beijing, China. The authors declare that this affiliation had no influence on the study design, data collection, analysis, interpretation, or the decision to publish, and that no other competing interests exist.

AI Use Statement

The authors declare that Deepseek-V3 was used solely for data analysis and language editing in the preparation of this manuscript. All scientific interpretations, results, and conclusions were determined by the authors.

Ethical Approval and Consent to Participate

Not applicable.

References

[1] Sun, Z., Xu, Y., Liu, Y., He, W., Kong, L., Wu, F., ... & Cui, L. (2024). A survey on federated recommendation systems. *IEEE Transactions on Neural Networks and Learning Systems*, *36*(1), 6-20. [CrossRef]

[2] Yin, H., Qu, L., Chen, T., Yuan, W., Zheng, R., Long, J., ... & Zhang, C. (2025). On-device recommender systems: A comprehensive survey. *Data Science and Engineering*, 1-30. [CrossRef]

[3] Zhang, C., Long, G., Zhang, Z., Li, Z., Zhang, H., Yang, Q., & Yang, B. (2025). Personalized recommendation models in federated settings: A survey. *IEEE Transactions on Knowledge and Data Engineering*. [CrossRef]

[4] Chai, D., Wang, L., Chen, K., & Yang, Q. (2020). Secure federated matrix factorization. *IEEE Intelligent Systems*, *36*(5), 11-20. [CrossRef]

[5] Perifanis, V., & Efraimidis, P. S. (2022). Federated neural collaborative filtering. *Knowledge-Based Systems*, *242*, 108441. [CrossRef]

[6] Zhang, H., Luo, F., Wu, J., He, X., & Li, Y. (2023). LightFR: Lightweight federated recommendation with privacy-preserving matrix factorization. *ACM Transactions on Information Systems*, *41*(4), 1-28. [CrossRef]

[7] Zhang, C., Long, G., Zhou, T., Yan, P., Zhang, Z., Zhang, C., & Yang, B. (2023). Dual personalization

on federated recommendation. In *Proceedings of the International Joint Conference on Artificial Intelligence* (pp. 4558–4566). [CrossRef]

[8] Jiang, J., Zhang, C., Zhang, H., Li, Z., Li, Y., & Yang, B. (2025, May). A Tutorial of Personalized Federated Recommender Systems: Recent Advances and Future Directions. In *Companion Proceedings of the ACM on Web Conference 2025* (pp. 21-24). [CrossRef]

[9] Ye, R., Ni, Z., Wu, F., Chen, S., & Wang, Y. (2023, July). Personalized federated learning with inferred collaboration graphs. In International conference on machine learning (pp. 39801-39817). PMLR.

[10] Zhang, C., Long, G., Zhou, T., Zhang, Z., Yan, P., & Yang, B. (2024). Gpfedrec: Graph-guided personalization for federated recommendation. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 4131–4142). [CrossRef]

[11] Li, Z., Long, G., & Zhou, T. (2023). Federated recommendation with additive personalization. *arXiv preprint arXiv:2301.09109*.

[12] Wang, X. (2025). UFGraphFR: An attempt at a federated recommendation system based on user text characteristics. *arXiv preprint arXiv:2501.08044*.

[13] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., & Chen, W. (2022). LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations*.

[14] Zhang, Q., Chen, M., Bukharin, A., Karampatziakis, N., He, P., Cheng, Y., ... & Zhao, T. (2023). Adalora: Adaptive budget allocation for parameter-efficient fine-tuning. *arXiv preprint arXiv:2303.10512*.

[15] Chen, J., Zhang, H., Li, H., Zhang, C., Li, Z., & Li, Y. (2025). Beyond Personalization: Federated Recommendation with Calibration via Low-rank Decomposition. *arXiv preprint arXiv:2506.09525*.

[16] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems, 30*.

[17] Kang, W. C., & McAuley, J. (2018). Self-Attentive Sequential Recommendation. In *2018 IEEE International Conference on Data Mining* (pp. 197–206). [CrossRef]

[18] Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., & Jiang, P. (2019). BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management* (pp. 1441–1450). [CrossRef]

[19] Zhou, T., Ma, Z., Wen, Q., Wang, X., Sun, L., & Jin, R. (2022, June). FEDformer: Frequency Enhanced Decomposed Transformer for Long-term Series Forecasting. In *International Conference on Machine Learning* (pp. 27268-27286). PMLR.

[20] Zhu, Y., Liu, J., Chowdhury, M., & Lai, F. (2024). Fedtrans: Efficient federated learning via multi-model transformation. *Proceedings of Machine Learning and Systems, 6*, 395-407.

[21] Ding, Y., Zhang, S., Fan, B., Sun, W., Liao, Y., & Zhou, P. Y. (2024). Fedloca: Low-rank coordinated adaptation with knowledge decoupling for federated recommendations. In *Proceedings of the ACM Conference on Recommender Systems* (pp. 690–700). [CrossRef]

[22] Harper, F. M., & Konstan, J. A. (2015). The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis), 5*(4), 1-19. [CrossRef]

[23] Cantador, I., Brusilovsky, P., & Kuflik, T. (2011). Second workshop on information heterogeneity and fusion in recommender systems (hetrec2011). In *Proceedings of the ACM Conference on Recommender Systems* (pp. 387–388). [CrossRef]

[24] Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer, 42*(8), 30–37. [CrossRef]

[25] He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T.-S. (2017). Neural collaborative filtering. In *Proceedings of the International World Wide Web Conference* (pp. 173–182). [CrossRef]

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

Xudong Wang received a Bachelor's degree in Computer Science and Engineering from Beijing University of Information Science and Technology in 2023 and pursued a Master's degree in Computer Science from Tianjin University of Technology.

Zhao Ruixin received a Bachelor's degree in Computer Science and Engineering fromBeijing University of Information Science and Technology in 2023. (Email: 2973124998@qq.com)