

Dynamic Feature-Embedding Communication via Codebook Distillation for Federated Recommendation

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

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

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

Jiankui Zhou
Fudan University
Shanghai, China
jkzhou24@m.fudan.edu.cn

Yaqiong Li
Fudan University
Shanghai, China
liyq22@m.fudan.edu.cn

Hansu Gu
Independent
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

Federated recommendation systems commonly protect user privacy by keeping user parameters on local devices, while exchanging item parameters for collaborative model training. However, such item parameters usually model items independently and suffer from both efficiency and effectiveness challenges, making communication costs grow with the item space and limiting cross-item generalization and robustness to noisy feedback. To address these limitations, we propose to model items via shared latent feature embeddings for communication. Residual Quantization (RQ) provides a natural way to instantiate this communication by representing each item with a short sequence of discrete code IDs, i.e., Semantic IDs (SIDs). However, directly applying centralized and static RQ-based recommendation to federated learning is non-trivial due to 1) private and biased historical interactions and 2) evolving collaborative information. We propose **RQFedRec**, an RQ-based federated recommendation framework for dynamic feature-embedding communication. To construct globally aligned codebooks without accessing private interactions, RQFedRec introduces an **information distillation module**. Each client first learns item embeddings that encode local collaborative information from private interactions, and then distills such information into feature-indexed codebooks under globally shared SIDs, making sparse and biased local signals more compatible with server aggregation. To adapt to evolving collaborative information, RQFedRec introduces a **self-refining SID update module** that dynamically refines global SID assignments from aggregated codebooks. Extensive experiments demonstrate that RQFedRec improves recommendation performance and

reduces communication costs without relying on semantic information, while further benefiting from public semantics when available. Code is available at <https://github.com/Mingzhe-Han/RQFedRec>.

CCS Concepts

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

Keywords

recommendation, federated learning, residual quantization

ACM Reference Format:

Mingzhe Han, Jiahao Liu, Dongsheng Li, Jiankui Zhou, Yaqiong Li, Hansu Gu, Peng Zhang, Ning Gu, and Tun Lu. 2026. Dynamic Feature-Embedding Communication via Codebook Distillation for Federated Recommendation. In *Proceedings of the 35th ACM International Conference on Information and Knowledge Management (CIKM '26)*, November 07–11, 2026, Rome, Italy. ACM, New York, NY, USA, 12 pages. <https://doi.org/10.1145/3799682.3841023>

*Corresponding author.

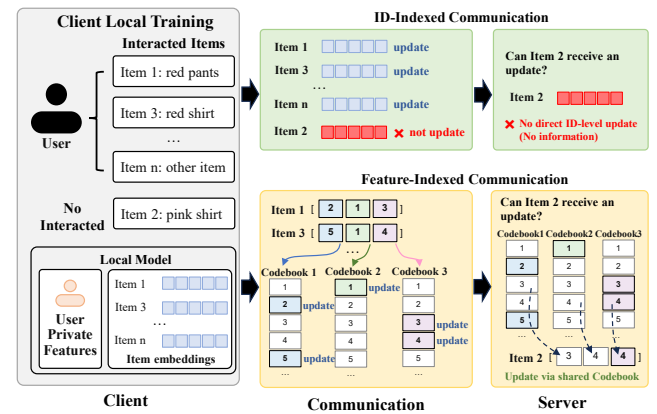


Figure 1: Illustration of two communication paradigms in federated learning.

1 Introduction

Recommendation algorithms [19, 20] aim to deliver personalized content from historical user interaction data [41, 42, 50]. However, user behaviors often contain sensitive personal information [4, 26, 49]. With the increasing enforcement of data protection regulations [34], it becomes difficult to directly collect raw interaction data on a central server. Federated learning [15, 38] offers a promising alternative by training models on distributed client devices and transmitting model information, thereby enabling recommendation while keeping user data local.

Existing federated recommendation algorithms [2, 6, 44] commonly rely on item information for communication. Specifically, due to the privacy risks of user preferences, user parameters are usually kept on local devices, while item parameters are communicated between clients and the server. In recommendation algorithms, these item parameters are mainly item embeddings [43], where each row is an embedding of a specific item ID [9]. Therefore, the communicated information consists of such ID-indexed embeddings; we refer to this setting as the **ID-indexed** communication paradigm. This paradigm is compatible with existing recommendation algorithms and easy to deploy in practical federated recommendation.

Despite its popularity, ID-indexed item communication is limited in federated recommendation in terms of both **efficiency** and **effectiveness**. For efficiency, this paradigm transmits item information through independent item embeddings. Thus, the communication cost increases with the number of items, making frequent transmission expensive in large-scale recommender systems [47]. For effectiveness, such communication updates item embeddings associated with isolated item IDs, which limits cross-item generalization and robustness to noisy feedback. As illustrated in Figure 1, if a user only interacts with Item 1 (red pants) and Item 3 (red shirt), the server only receives updates for these two item IDs. Although Item 2 (pink shirt) may be related to the interacted items, it is not involved in this communication. Therefore, this user behavior cannot increase the model’s preference estimate for Item 2 through the uploaded item updates. Moreover, when a user accidentally clicks an item, the noisy feedback is directly reflected in its corresponding item embedding. This item-level update can easily bias the representation of the item, especially when the item is long-tailed and has only a few interactions [23].

These limitations motivate a **feature-indexed** communication paradigm, where item information is communicated through shared latent features from collaborative information instead of item embeddings indexed by isolated item IDs. Recent Residual Quantization (RQ)-based methods [14, 29] provide a possible tool: an item can be represented by a short sequence of discrete IDs, where each ID corresponds to a latent feature and can be shared by multiple items. Such an ID sequence is usually referred to as the Semantic ID (SID), and each position in the SID corresponds to a codebook, which stores the code embeddings (i.e., the embeddings selected by SIDs at this position). For the efficiency limitation, the communicated parameters are determined by the size of codebooks rather than the number of item IDs, and new items can be assigned to existing codebooks without introducing a new independent item embedding. For the effectiveness limitation, related items that the user has not

interacted with can still receive updates through shared codebooks. As illustrated in Figure 1, Item 2 can also be updated because the user interacted with Item 3, which shares part of the codebook embedding with Item 2. Moreover, each code embedding aggregates information from multiple items, making the communication less sensitive to isolated noisy feedback.

However, RQ-based recommendation cannot be directly combined with federated learning. Existing RQ-based methods usually construct SIDs and codebooks from item semantics or collaborative information learned from centralized historical interactions [51]. The generated SIDs are then used as static item features for downstream recommendation model training. This centralized and static pipeline raises two challenges under federated constraints. **First, private and biased interactions make centralized SID construction difficult.** Constructing centralized SIDs requires user interaction data to capture collaborative information, but such data are private and cannot be directly uploaded. Moreover, sparse and biased local interactions make it difficult to construct globally aligned SIDs and codebooks. **Second, federated training makes static SIDs insufficient.** Federated recommendation is an iterative process where collaborative information is continuously encoded into item embeddings. Thus, SIDs and codebooks constructed before training may become misaligned with the updated collaborative information, making dynamic refinement necessary.

Motivated by the above observations, we propose RQFedRec, an RQ-based federated recommendation framework for **feature-indexed** communication. RQFedRec replaces the upload of ID-indexed item embeddings with compact feature-indexed codebooks while keeping the local recommendation backbone (e.g., MF) unchanged. In each communication round, clients upload local codebooks to the server for aggregation, and the aggregated codebooks are then broadcast back for the next round of local training. Rather than simply combining RQ-based recommendation with federated learning, RQFedRec introduces two key designs. First, the **information distillation module** addresses the difficulty of learning globally aligned codebooks from private and biased interactions. Instead of directly optimizing codebooks with sparse local recommendation loss, each client first learns item embeddings as flexible local representations to capture collaborative information, and then distills such information into local codebooks under shared SIDs. This aligns locally learned collaborative information for server aggregation without uploading raw interactions. Second, the **self-refining SID update module** addresses the limitation of static SIDs. Rather than keeping SIDs fixed throughout training, the server periodically refines SIDs based on aggregated codebooks, allowing global SIDs to adapt to the updated collaborative information during federated training.

Our main contributions are summarized as follows:

- We propose **feature-indexed** communication to address the limitations of conventional ID-indexed item communication in federated recommendation. Instead of transmitting item embeddings tied to isolated item IDs, feature-indexed communication transmits shared feature embeddings that can be reused by multiple items.
- We instantiate feature-indexed communication with Residual Quantization and propose **RQFedRec**. We design an

information distillation module to transfer locally learned collaborative information into shared codebooks, together with a self-refining SID update module to adapt the feature-indexed structure during federated learning.

- We conduct extensive experiments on multiple real-world datasets. The results show that RQFedRec improves recommendation performance and communication efficiency without relying on semantic priors, while further benefiting from public semantics when available.

2 Related Work

2.1 Federated Learning

Federated Learning (FL) [40] is a distributed learning framework that enables multiple clients to collaboratively train a shared model without the collection of raw data. FedAvg [25] aggregates client updates by weighted averaging according to the data size of each client. Building upon FedAvg, many studies focus on improving server aggregation [36] and stabilizing client optimization [16] under statistical heterogeneity. In addition, some FL methods focus on client selection [13] to improve stability and update sparsification [1] to reduce the communication cost. Despite these advances, those FL algorithms primarily target supervised learning and do not explicitly address recommendation scenarios, where parameters are linearly related to the number of items and interactions are highly sensitive to noise.

2.2 Federated Recommendation Algorithms

Federated recommendation [31, 32, 46] adapts the FL framework to recommendation algorithms. Early work such as FedRec [18] masks user interactions by sampling unrated items and using virtual ratings to reduce privacy leakage. FedMF [2] further protects gradients via secure aggregation such as homomorphic encryption. Subsequent methods like FedNCF [28] extend the idea to neural collaborative filtering.

To improve personalization in federated recommendation, recent studies explore personalized federated learning mechanisms. PFedRec [43] introduces a dual personalization strategy to optimize both global and client models. GPFedRec [44] leverages user relationship graphs to capture inter-user correlations. FedRAP [17] enhances personalization by decomposing item embeddings into a global component learned via federation and a local additive component learned on each client. FedCIA [6] emphasizes the aggregation of collaborative information rather than directly averaging item parameters. While these approaches enable privacy-preserving training, they often upload ID-indexed item embeddings for aggregation, which scales poorly with the number of items, generalizes weakly across related items, and is sensitive to noisy feedback [5].

2.3 RQ-based Recommendation

Recent generative recommendation studies have explored residual-quantized codebooks and Semantic IDs (SIDs) to represent items as short discrete token sequences [29, 33]. Such discrete IDs have attracted increasing attention in generative recommendation, since they reduce the item output space and make the recommendation compatible with sequence modeling architectures. However, existing RQ-based recommendation methods [37] are mainly designed

for centralized and static recommendation rather than federated communication. They usually assume that historical interactions or collaborative item embeddings are available for SID construction, and the constructed SIDs remain fixed during downstream training [39]. This assumption does not hold in federated recommendation, where interactions are private and distributed, and collaborative item information evolves during iterative training. In addition, FedMM [48] encodes user and item information into a codebook for the CTR task, which is not suitable for our top-N retrieval setting and ignores dynamic collaborative information in federated training. As a result, SIDs learned by existing methods cannot be directly used as a globally aligned and dynamically adaptive communication structure in federated recommendation.

3 Preliminaries

3.1 Recommendation Algorithms

Given a user set $\mathcal{U}$ and an item set $\mathcal{I}$, a recommendation algorithm aims to predict user-item interactions based on an interaction dataset $\mathcal{D} = \{(u, i, y_{ui})\}$, where (u, i) is a user-item pair and $y_{ui} \in \{0, 1\}$ indicates whether user u interacted with item i .

Most recommendation algorithms can be represented as a prediction function $R(\cdot)$ that outputs a prediction score $\hat{y}_{ui}$:

$$\hat{y}_{ui} = R(u, i) = P(E_u(u; \theta_u), E_i(i; \theta_i); \theta_p), \quad (1)$$

where $E_u(\cdot; \theta_u)$ and $E_i(\cdot; \theta_i)$ are the user and item embedding layers with parameters θ_u and θ_i , respectively, and $P(\cdot; \theta_p)$ is the prediction function, such as a dot product or MLP layer parameterized by θ_p . We denote all model parameters as $\theta = \{\theta_u, \theta_i, \theta_p\}$.

3.2 Federated Recommendation Algorithms

In federated recommendation, user-item interactions are stored locally to protect user privacy. We assume there are K clients or devices, indexed by $k \in \{1, \dots, K\}$, and client k holds a local dataset $\mathcal{D}_k \subset \mathcal{D}$. The goal of federated recommendation is to learn a global recommendation model without collecting user-item interactions on the server. In conventional federated learning methods such as FedAvg [25], each client k first trains its local parameters θ_k based on its local dataset $\mathcal{D}_k$, and the server aggregates client models by weighted averaging:

$$\theta^{t+1} = \sum_{k=1}^K \frac{|\mathcal{D}_k|}{\sum_{j=1}^K |\mathcal{D}_j|} \theta_k^{t+1}, \quad (2)$$

where θ^{t+1} denotes the global parameters after aggregation in round t .

In federated recommendation, directly uploading all model parameters may expose private user information. A common practice is partial parameter upload [18], where user parameters θ_u are kept on local devices, while the server only aggregates item parameters or public parameters. Since the uploaded item parameters are usually organized according to item IDs, we refer to this setting as ID-indexed communication.

3.3 Residual Quantization k-means

Residual Quantization k-means (RQ-Kmeans) [3] is a multi-level vector quantization method that represents each embedding by a sequence of discrete code IDs. Given an item embedding table

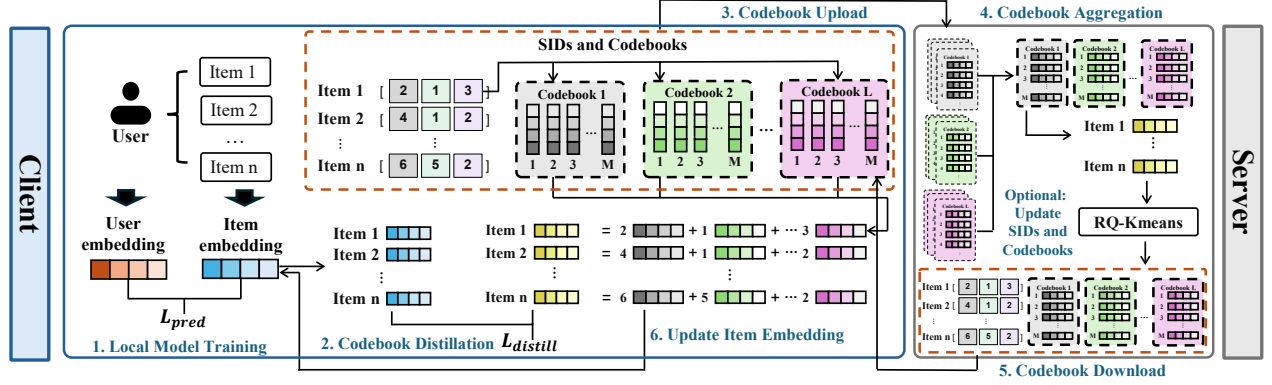


Figure 2: Overview of the RQFedRec framework. We only illustrate a single client for simplicity.

$\mathbf{E} = \{\mathbf{e}_i\}_{i \in \mathcal{I}}$, where $\mathbf{e}_i \in \mathbb{R}^d$, RQ-Kmeans generates L codebooks $\{\mathcal{B}^{(\ell)}\}_{\ell=1}^L$. Each codebook $\mathcal{B}^{(\ell)} = \{\mathbf{b}_1^{(\ell)}, \dots, \mathbf{b}_M^{(\ell)}\} \subset \mathbb{R}^d$ contains M code embeddings. Each item is encoded into a length- L tuple of discrete code IDs:

$$\mathbf{q}_i = (q_i^{(1)}, \dots, q_i^{(L)}), \quad q_i^{(\ell)} \in \{1, \dots, M\}. \quad (3)$$

We refer to the code ID tuple $\mathbf{q}_i$ as the Semantic ID (SID) of item i . Given the SID $\mathbf{q}_i$, the item embedding $\mathbf{e}_i$ can be approximately reconstructed by summing the selected code embeddings:

$$\tilde{\mathbf{e}}_i = \sum_{\ell=1}^L \mathbf{b}_{q_i^{(\ell)}}^{(\ell)}. \quad (4)$$

RQ-Kmeans encodes each item embedding sequentially by quantizing the residual left by previous codebooks. Specifically, it takes the original embedding $\mathbf{e}_i$ as the first-layer residual: $\mathbf{r}_i^{(1)} = \mathbf{e}_i$. At the ℓ -th layer, k-means is applied to the residual embeddings $\{\mathbf{r}_i^{(\ell)}\}_{i \in \mathcal{I}}$, and the cluster centers are used as the code embeddings $\{\mathbf{b}_1^{(\ell)}, \dots, \mathbf{b}_M^{(\ell)}\}$ in codebook $\mathcal{B}^{(\ell)}$. Each item i is assigned to the nearest code embedding:

$$q_i^{(\ell)} = \arg \min_{j \in \{1, \dots, M\}} \|\mathbf{r}_i^{(\ell)} - \mathbf{b}_j^{(\ell)}\|_2. \quad (5)$$

Then, the residual embedding for the next layer is updated as:

$$\mathbf{r}_i^{(\ell+1)} = \mathbf{r}_i^{(\ell)} - \mathbf{b}_{q_i^{(\ell)}}^{(\ell)}. \quad (6)$$

After L layers, RQ-Kmeans obtains a SID $\mathbf{q}_i$ for each item and a set of codebooks for reconstructing item embeddings.

4 Methodology

4.1 Overview

In this section, we introduce RQFedRec, an RQ-based federated recommendation framework that replaces ID-indexed item embedding communication with **feature-indexed** codebook communication. Instead of uploading item embeddings directly, RQFedRec represents each item by a SID sequence and communicates the shared codebooks associated with these SIDs. Figure 2 illustrates one communication round of RQFedRec.

Each client first trains its local recommendation model on private interactions. After local training, the **information distillation module** distills the collaborative information captured by the learned item embeddings into local codebooks under the current SIDs. Then, each client uploads its local codebooks to the server. The server aggregates the uploaded codebooks to obtain updated global codebooks. After every τ communication rounds, the **self-refining SID update module** reconstructs global item embeddings from the aggregated codebooks and updates SIDs and codebooks. Finally, the updated codebooks and SIDs are broadcast to clients for the next round, where clients refresh their local item embeddings and continue training.

4.2 SID and Codebook Initialization

Before federated training, RQFedRec needs to initialize globally shared SIDs and global codebooks for feature-indexed communication. The global SIDs specify how each item selects feature embeddings from codebooks, while the global codebooks store the shared feature embeddings communicated between clients and the server. After initialization, the server broadcasts the SIDs and codebooks to all clients. In this way, all clients learn and upload codebooks under the same index space, which enables consistent aggregation of the uploaded codebooks.

Formally, let the global codebooks at round t be

$$\mathcal{B}^t = \{\mathbf{B}^{t,(\ell)} \in \mathbb{R}^{M \times d}\}_{\ell=1}^L, \quad (7)$$

where L is the number of RQ layers, M is the number of code embeddings in each codebook, and d is the embedding dimension. Each item i is assigned a length- L SID:

$$\mathbf{q}_i^t = (q_i^{t,(1)}, \dots, q_i^{t,(L)}), \quad q_i^{t,(\ell)} \in \{1, \dots, M\}. \quad (8)$$

Given the global codebooks and the SID of item i , its embedding can be reconstructed as $\tilde{\mathbf{v}}_i^t = \sum_{\ell=1}^L \mathbf{B}^{t,(\ell)}[\mathbf{q}_i^{t,(\ell)}]$, where $\mathbf{B}^{t,(\ell)}[\mathbf{q}_i^{t,(\ell)}]$ denotes the selected code embedding from the ℓ -th codebook.

The global codebooks are randomly initialized in the collaborative embedding space:

$$\mathbf{B}^{0,(\ell)}[m] \sim \mathcal{N}(\mathbf{0}, \sigma_b^2 \mathbf{I}), \quad \ell = 1, \dots, L, \quad m = 1, \dots, M, \quad (9)$$

For SIDs, RQFedRec supports semantic initialization and random initialization. When public item content features are available, we

obtain an initial semantic item embedding $\mathbf{s}_i \in \mathbb{R}^{d_s}$ for each item i from these public features. The server applies RQ-Kmeans to the semantic embeddings $\{\mathbf{s}_i\}_{i \in \mathcal{I}}$ and retains the resulting discrete code assignments as the initial global SIDs $\mathbf{q}^0$. Semantic SID initialization provides a useful starting structure for federated communication, especially when collaborative information is insufficient in early training.

However, in many recommendation scenarios, reliable semantic information may be unavailable or noisy. Moreover, using external semantic features may make the comparison with existing federated recommendation baselines unfair. Therefore, RQFedRec also supports randomly initializing SIDs without any external semantic information:

$$q_i^{0,(\ell)} \sim \text{Uniform}\{1, \dots, M\}, \quad i \in \mathcal{I}, \quad \ell = 1, \dots, L. \quad (10)$$

This setting allows RQFedRec to be trained purely from interaction data and helps to verify that the proposed feature-indexed communication framework does not rely on external semantic information.

It is worth noting that both SID initialization strategies only provide the starting assignments for feature-indexed communication. During federated training, the codebooks are updated by the information distillation module, and the SIDs are periodically refined by the self-refining SID update module. Thus, the SIDs and codebooks can gradually adapt to collaborative information learned from clients regardless of the SID initialization strategy.

4.3 Communication Pipeline

In this section, we describe the training pipeline in a single communication round. For simplicity, we omit the round superscript t when there is no ambiguity.

4.3.1 Local Model Training. Before local training in the current round, each client has received the global codebooks $\mathcal{B}$ and SIDs $\mathbf{q}$ broadcast by the server, and initializes its local item embeddings accordingly. Client k then trains its local recommendation model on its private dataset $\mathcal{D}_k$ by minimizing a prediction loss:

$$\mathcal{L}_{\text{pred}}^{(k)}(\theta_k) = \frac{1}{|\mathcal{D}_k|} \sum_{(u,i,y) \in \mathcal{D}_k} \ell_{\text{pred}}(R(u,i), y), \quad (11)$$

where ℓ_{pred} can be instantiated as common recommendation objectives, such as binary cross-entropy or pairwise ranking loss. After local training, we denote the learned item embedding of item i on client k as $\mathbf{e}_i^{(k)}$.

4.3.2 Information Distillation Module. In RQFedRec, all clients share the same global SIDs and codebooks for feature-indexed communication. However, each client observes collaborative information from its own local interactions, which can be sparse, biased, and different from those of other clients [21]. If we directly train the codebooks with sparse local interaction loss, such biased local information may be forced into the current SID groups and mixed with unrelated items. This makes codebook optimization unstable and may harm the alignment required for server aggregation.

To address this issue, RQFedRec introduces an **information distillation module**. Instead of directly learning collaborative information on the codebooks, each client first trains item embeddings through local model training. Then, each client distills the

useful collaborative information captured by the learned item embeddings into a local set of feature-indexed codebooks under the globally shared SIDs. In this way, sparse and biased local information is represented in the same SID space, allowing the server to aggregate the uploaded codebooks.

Specifically, client k maintains a local codebook

$$\mathcal{B}_k = \left\{ \mathbf{B}_k^{(\ell)} \in \mathbb{R}^{M \times d} \right\}_{\ell=1}^L. \quad (12)$$

Given the current global SIDs $\mathbf{q}$, the embedding reconstructed from the local codebooks is: $\tilde{\mathbf{e}}_i^{(k)} = \sum_{\ell=1}^L \mathbf{B}_k^{(\ell)} [q_i^{(\ell)}]$.

The locally trained item embedding $\mathbf{e}_i^{(k)}$ acts as the teacher, and the reconstructed embedding $\tilde{\mathbf{e}}_i^{(k)}$ acts as the student. Client k optimizes its local codebook by minimizing the reconstruction loss:

$$\mathcal{L}_{\text{distill}}^{(k)}(\mathcal{B}_k) = \frac{1}{|\mathcal{I}_k|} \sum_{i \in \mathcal{I}_k} \left\| \mathbf{e}_i^{(k)} - \tilde{\mathbf{e}}_i^{(k)} \right\|_2^2, \quad (13)$$

where $\mathcal{I}_k = \{i \mid (u, i, y) \in \mathcal{D}_k\}$ is the set of items appearing in client k 's local dataset. During this stage, the locally trained item embeddings and the global SID assignment are fixed, and only the local codebook $\mathcal{B}_k$ is optimized. After information distillation, client k uploads $\mathcal{B}_k$ to the server for aggregation.

4.3.3 Server Codebook Aggregation. After receiving local codebooks from clients, the server aggregates them by weighted averaging. For each RQ layer $\ell \in \{1, \dots, L\}$, the global codebook is computed as:

$$\mathbf{B}^{(\ell)} = \sum_{k=1}^K \frac{|\mathcal{D}_k|}{\sum_{j=1}^K |\mathcal{D}_j|} \mathbf{B}_k^{(\ell)}. \quad (14)$$

The aggregated codebooks $\mathcal{B}$ summarize collaborative information distilled from different clients. Since L , M , and d are fixed, the upload size of RQFedRec is bounded and does not grow with the number of items.

4.3.4 Local Embedding Update. After codebook aggregation, the server broadcasts the updated global codebooks $\mathcal{B}$ and SIDs $\mathbf{q}$ to clients. Each client then refreshes its local item embeddings by reconstructing them from the received codebooks:

$$\mathbf{e}_i^{(k)} \leftarrow \sum_{\ell=1}^L \mathbf{B}^{(\ell)} [q_i^{(\ell)}]. \quad (15)$$

The refreshed item embeddings are used as the initialization for the next round of local model training. This update allows each client to receive item information aggregated from other clients through shared feature-indexed codebooks.

4.3.5 Self-Refining SID Update Module. The information distillation module transfers locally learned collaborative information into the shared codebook space, enabling the server to aggregate client information under globally aligned SIDs. However, this only solves the aggregation problem under the static SID assignment. Different from conventional RQ-based recommendation, where SIDs are usually generated before training and then kept fixed, the item embeddings in federated recommendation continuously evolve through iterative local training and server aggregation. As clients update item embeddings from private interactions and the

server aggregates distilled codebooks across rounds, the collaborative structure reflected by the global codebooks may gradually change. Therefore, a static SID assignment may become misaligned with the updated item embeddings and limit the effectiveness of feature-indexed communication.

To address this issue, RQFedRec introduces a **self-refining SID update module**. The key idea is to periodically refine SIDs according to the federated item embeddings reconstructed from the aggregated codebooks.

Specifically, the server reconstructs a global embedding for each item i using the current global codebooks and SIDs:

$$\bar{\mathbf{e}}_i = \sum_{\ell=1}^L \mathbf{B}^{(\ell)} [\mathbf{q}_i^{(\ell)}]. \quad (16)$$

Here, $\bar{\mathbf{e}}_i$ is not uploaded by any client, but reconstructed on the server from the aggregated codebooks. It summarizes the collaborative information distilled from different clients under the current SID assignment.

After every τ communication rounds, the server applies RQ-Kmeans to the reconstructed item embeddings $\{\bar{\mathbf{e}}_i\}_{i \in \mathcal{I}}$:

$$(\mathbf{q}^{\text{new}}, \mathcal{B}^{\text{new}}) = \text{RQ-Kmeans}(\{\bar{\mathbf{e}}_i\}_{i \in \mathcal{I}}). \quad (17)$$

If the current round is not a SID update round, we keep $\mathbf{q}^{\text{new}} = \mathbf{q}$ and $\mathcal{B}^{\text{new}} = \mathcal{B}$.

This module makes SIDs dynamic during federated training rather than static preprocessing tokens. By periodically refining SIDs based on aggregated codebooks, RQFedRec allows the SIDs and codebooks to gradually adapt to the updated collaborative information without collecting raw interactions or uploading raw local item embeddings.

4.4 Algorithm

Our complete training procedure is summarized in Algorithm 1.

4.5 Privacy Protection

RQFedRec focuses on changing the communication paradigm of federated recommendation from ID-indexed communication to feature-indexed communication. Therefore, it is compatible with the privacy protection mechanisms used by existing ID-indexed federated recommendation methods. Specifically, if a base method applies privacy protection to item embeddings or item updates, RQFedRec can perform codebook distillation on the protected item embeddings and upload the codebooks under the same protection setting.

In this sense, RQFedRec does not require additional access to raw interactions, user embeddings, or unprotected item embeddings. It only replaces the communicated object from item embeddings to feature-indexed codebooks. Thus, when the original ID-indexed communication satisfies a given privacy requirement, RQFedRec can inherit the same privacy setting by learning and aggregating codebooks from the protected item information.

4.6 Communication Cost

We compare the communication cost of RQFedRec with ID-indexed communication methods such as *FedMF* [2]. In ID-indexed communication, each client uploads and downloads item embeddings

Algorithm 1 RQFedRec: Feature-Indexed Federated Recommendation

Require: Clients $\{1, \dots, K\}$ with local datasets $\{\mathcal{D}_k\}$; codebook size M ; RQ layers L ; SID update period τ ; total rounds T .

- 1: Randomly initialize global codebooks $\mathcal{B}^0$, and initialize global SIDs $\mathbf{q}^0$ randomly or with optional semantic information.
- 2: **for** round $t = 0, 1, \dots, T - 1$ **do**
- 3: **for** each client k in parallel **do**
- 4: **Local training:** update local model parameters θ_k on $\mathcal{D}_k$ by minimizing Eq. (11).
- 5: **Information distillation:** optimize local codebooks $\mathcal{B}_k^t$ under global SIDs $\mathbf{q}^t$ by minimizing Eq. (13).
Upload local codebooks $\mathcal{B}_k^t$ to the server.
- 6: **end for**
- 7: **Codebook aggregation:** aggregate uploaded codebooks to obtain $\mathcal{B}^{t+1}$ by Eq. (14).
- 8: **if** $(t + 1) \bmod \tau = 0$ **then**
- 9: Reconstruct global item embeddings from $\mathcal{B}^{t+1}$ and $\mathbf{q}^t$ by Eq. (16).
- 10: **Self-refining SID update:** regenerate SIDs $\mathbf{q}^{t+1}$ and codebooks $\mathcal{B}^{t+1}$ by RQ-Kmeans using Eq. (17).
- 11: **else**
- 12: Keep $\mathbf{q}^{t+1} = \mathbf{q}^t$.
- 13: **end if**
- 14: Server broadcasts $\mathcal{B}^{t+1}$ and $\mathbf{q}^{t+1}$ for the next round.
- 15: Each client reconstructs local item embeddings from $\mathcal{B}^{t+1}$ and $\mathbf{q}^{t+1}$.
- 16: **end for**

organized by item IDs. Since the item embedding table has size $n_i \times d$, where n_i is the number of items and d is the embedding dimension, the communication cost scales as $O(n_i d)$.

In contrast, RQFedRec communicates feature-indexed codebooks instead of full item embedding tables. With L RQ layers and M code embeddings in each codebook, the codebook communication cost scales as $O(LMd)$. When a SID update is performed, the server additionally broadcasts the updated SID assignment, whose cost is $O(n_i L)$. The download cost of RQFedRec becomes $O(LMd + n_i L)$. Because L and M are fixed hyperparameters and $L \ll d$, the SID table is much cheaper than transmitting full item embeddings. In our experiments, we set $LM \ll n_i$, so the total download cost $O(LMd + n_i L)$ is consistently smaller than that of the ID-indexed item embedding table. Table 3 provides a comparison of the download cost.

5 Theoretical Analysis

We provide a concise analysis to explain why feature-indexed communication can reduce the influence of noisy updates. This proof leads to an intuitive conclusion: independent item embeddings are less resistant to noise than feature embeddings shared by multiple items. This is because ID-indexed aggregation updates each item embedding only with independent item signals, while feature-indexed aggregation pools signals from all items sharing the same feature embedding. We show that this feature-indexed aggregation can strictly reduce the variance of noisy updates.

5.1 Noise Reduction by Feature-Indexed Aggregation

Consider a set of items $\mathcal{I}$. In ID-indexed communication, each item i is treated as an independent communication unit. Let $\mathcal{K}(i)$ denote the set of clients that contribute updates for item i , and let $n_i = |\mathcal{K}(i)|$. Assume that client k observes a noisy item update:

$$\mathbf{z}_i^{(k)} = \mathbf{z}_i^* + \boldsymbol{\epsilon}_i^{(k)}, \quad \mathbb{E}[\boldsymbol{\epsilon}_i^{(k)}] = \mathbf{0}, \quad \mathbb{E}\|\boldsymbol{\epsilon}_i^{(k)}\|_2^2 = \sigma^2, \quad (18)$$

where $\mathbf{z}_i^*$ is the ideal item embedding without noise information, and $\boldsymbol{\epsilon}_i^{(k)}$ denotes noise caused by sparse interactions, stochastic optimization, or noisy feedback.

The ID-indexed estimator for item i is:

$$\hat{\mathbf{z}}_i^{\text{ID}} = \frac{1}{n_i} \sum_{k \in \mathcal{K}(i)} \mathbf{z}_i^{(k)}. \quad (19)$$

In feature-indexed communication, items are mapped to shared feature indices. Let $q_i \in \{1, \dots, M\}$ denote the feature index of item i , and let $\mathcal{I}_m = \{i \in \mathcal{I} \mid q_i = m\}$ be the set of items sharing feature index m . Thus, items in $\mathcal{I}_m$ share the same underlying feature embedding $\mathbf{b}_m^*$. For each item $i \in \mathcal{I}_m$, its client update can be written as:

$$\mathbf{z}_i^{(k)} = \mathbf{b}_m^* + \boldsymbol{\epsilon}_i^{(k)}. \quad (20)$$

The feature-indexed estimator aggregates all updates assigned to feature index m :

$$\hat{\mathbf{b}}_m^{\text{F}} = \frac{1}{N_m} \sum_{i \in \mathcal{I}_m} \sum_{k \in \mathcal{K}(i)} \mathbf{z}_i^{(k)}, \quad N_m = \sum_{i \in \mathcal{I}_m} n_i. \quad (21)$$

THEOREM 5.1. *Assume that the noise terms are independent and zero-mean with variance σ^2 . For an item i with feature index $q_i = m$, let $\mathcal{E}_{\text{ID}}(i)$ denote the expected aggregation error under ID-indexed aggregation, and let $\mathcal{E}_{\text{F}}(m)$ denote the expected aggregation error under feature-indexed aggregation. Then,*

$$\mathcal{E}_{\text{F}}(m) \leq \mathcal{E}_{\text{ID}}(i). \quad (22)$$

The inequality is strict when the feature index m is shared by at least one additional item with client updates.

PROOF. According to the ID-indexed estimator, its expected aggregation error is:

$$\begin{aligned} \mathcal{E}_{\text{ID}}(i) &= \mathbb{E} \|\hat{\mathbf{z}}_i^{\text{ID}} - \mathbf{z}_i^*\|_2^2 \\ &= \mathbb{E} \left\| \frac{1}{n_i} \sum_{k \in \mathcal{K}(i)} \boldsymbol{\epsilon}_i^{(k)} \right\|_2^2. \end{aligned} \quad (23)$$

Since the noise terms are independent and zero-mean, we have:

$$\mathcal{E}_{\text{ID}}(i) = \frac{1}{n_i^2} \sum_{k \in \mathcal{K}(i)} \mathbb{E} \|\boldsymbol{\epsilon}_i^{(k)}\|_2^2 = \frac{\sigma^2}{n_i}. \quad (24)$$

Similarly, according to the feature-indexed estimator, its expected aggregation error is:

$$\begin{aligned} \mathcal{E}_{\text{F}}(m) &= \mathbb{E} \|\hat{\mathbf{b}}_m^{\text{F}} - \mathbf{b}_m^*\|_2^2 \\ &= \mathbb{E} \left\| \frac{1}{N_m} \sum_{j \in \mathcal{I}_m} \sum_{k \in \mathcal{K}(j)} \boldsymbol{\epsilon}_j^{(k)} \right\|_2^2. \end{aligned} \quad (25)$$

Table 1: Statistics of our datasets.

datasets	# Users	# Items	# Interactions
ML-100K	943	1682	100000
ML-1M	6040	3706	1000209
Steam	23310	5237	525922
Movies	16641	20000	130342
VideoGames	13107	20000	112043

By independence and zero mean, we obtain:

$$\mathcal{E}_{\text{F}}(m) = \frac{1}{N_m^2} \sum_{j \in \mathcal{I}_m} \sum_{k \in \mathcal{K}(j)} \mathbb{E} \|\boldsymbol{\epsilon}_j^{(k)}\|_2^2 = \frac{\sigma^2}{N_m}. \quad (26)$$

Since item i belongs to $\mathcal{I}_m$, the feature index m aggregates at least the updates of item i :

$$N_m = \sum_{j \in \mathcal{I}_m} n_j \geq n_i. \quad (27)$$

Therefore,

$$\mathcal{E}_{\text{F}}(m) = \frac{\sigma^2}{N_m} \leq \frac{\sigma^2}{n_i} = \mathcal{E}_{\text{ID}}(i). \quad (28)$$

The equality holds when $N_m = n_i$. When feature index m is shared by at least one additional item with client updates, we have $N_m > n_i$, and thus:

$$\mathcal{E}_{\text{F}}(m) < \mathcal{E}_{\text{ID}}(i). \quad (29)$$

□

Theorem 5.1 shows a core advantage of feature-indexed communication. ID-indexed aggregation averages noisy updates only within the independent item ID, while feature-indexed aggregation pools updates from all items sharing the same feature index. Therefore, the effective number of observations increases from n_i to N_m , reducing the noise variance from σ^2/n_i to σ^2/N_m . This explains why feature-indexed communication can improve robustness to noisy and sparse ID-indexed feedback.

6 Experiments

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

- **RQ1:** Does RQFedRec outperform representative federated recommendation baselines in recommendation accuracy?
- **RQ2:** Do our information distillation module and self-refining SID update module contribute to the performance of RQFedRec?
- **RQ3:** Can feature-indexed communication address the efficiency and effectiveness limitations of ID-indexed communication?
- **RQ4:** Is the feature-indexed codebook construction in RQFedRec more effective than directly using semantic information or alternative codebook construction methods?

6.1 Experimental Settings

6.1.1 Datasets. Our experiments are conducted on five widely used datasets: ML-100K [7], ML-1M [7], Steam [27], Movies [24] and VideoGames [24]. These datasets are split into training and test

Table 2: Overall comparison on five datasets. The boldface indicates the best result and the underline indicates the second-best.

Dataset	Metric	No Agg.	ID-Indexed Aggregation							Ours	
		MF	FedMF	FedNCF	PFedRec	GPFedRec	FedRAP	FedCIA	FedCA	RQFedRec-R	RQFedRec-S
ML-100K	Recall@10	0.0756	0.1813	0.1680	0.1681	0.1162	0.0600	0.1903	0.1184	<u>0.1928</u>	0.1973
	MRR@10	0.3232	0.5501	0.5568	0.5685	0.4550	0.0600	0.5855	0.4107	<u>0.5973</u>	0.6168
	NDCG@10	0.1504	0.3198	0.3170	0.3233	0.2395	0.1260	0.3440	0.2008	<u>0.3533</u>	0.3650
ML-1M	Recall@10	0.0641	0.1285	0.1298	0.1177	0.0798	0.0592	0.1392	0.0806	<u>0.1497</u>	0.1517
	MRR@10	0.3601	0.5664	0.5590	0.5367	0.4269	0.3082	0.5947	0.4167	<u>0.6176</u>	0.6300
	NDCG@10	0.1816	0.3254	0.3265	0.3192	0.2396	0.1572	0.3611	0.1989	<u>0.3750</u>	0.3813
Steam	Recall@10	0.0652	0.0611	0.0535	0.0492	0.0562	0.0512	0.0752	0.0538	<u>0.0779</u>	0.0809
	MRR@10	0.1317	0.0913	0.0737	0.0799	0.0906	0.1095	0.1143	0.1068	0.1157	<u>0.1189</u>
	NDCG@10	0.0649	0.0508	0.0429	0.0429	0.0489	0.0522	0.0644	0.0528	<u>0.0653</u>	0.0673
Movies	Recall@10	0.1110	0.1194	0.1216	0.0858	0.1158	0.0531	0.1146	0.1018	0.1460	<u>0.1349</u>
	MRR@10	0.0996	0.0903	0.0914	0.0571	0.0881	0.0576	0.0894	0.0812	0.1117	<u>0.1057</u>
	NDCG@10	0.0804	0.0781	0.0799	0.0521	0.0760	0.0426	0.0768	0.0686	0.0978	<u>0.0913</u>
VideoGames	Recall@10	0.0163	<u>0.0414</u>	0.0198	0.0128	0.0312	0.0018	0.0300	0.0193	0.0483	0.0405
	MRR@10	0.0200	<u>0.0323</u>	0.0156	0.0097	0.0282	0.0019	0.0227	0.0199	0.0395	0.0290
	NDCG@10	0.0139	<u>0.0270</u>	0.0127	0.0081	0.0219	0.0014	0.0190	0.0147	0.0323	0.0259

sets in an 8:2 ratio, with 10% of the training set used as a validation set. Following a general federated recommendation setting [6], we do not restrict each client to a single user. A client can represent a user device or a user group. This setting includes the single user case as a special case and better matches our focus on communication under large item catalogs. Therefore, we partition each dataset into 100 clients and evaluate all methods under the same protocol. The statistics of these datasets are shown in Table 1.

6.1.2 Compared Methods. We compare two popular federated recommendation algorithms, FedMF [2] and FedNCF [28], and five methods that focus on user personalization, PFedRec [43], GPFedRec [44], FedRAP [17], FedCIA [6] and FedCA [45].

Our primary goal is to improve model performance within the same communication budget. Therefore, we mainly compare state-of-the-art (SOTA) methods rather than methods specifically designed for communication efficiency.

6.1.3 Evaluation Metrics. We use Recall [10], Mean Reciprocal Rank (MRR) [35] and Normalized Discounted Cumulative Gain (NDCG) [12] as the primary evaluation metrics. These are popular metrics in the federated recommendation scenario. We set $K = 10$ for these metrics.

6.1.4 Implementation Details. Our method supports two SID initialization strategies. The first is **random initialization**, denoted as **RQFedRec-R**, where SIDs are randomly initialized without using any external semantic information. The second is **semantic initialization**, denoted as **RQFedRec-S**, where public semantic information is used to initialize SIDs.

For semantic initialization, following existing large language model-based recommendation work [22, 30], we obtain a semantic embedding for each item by first generating a structured item feature profile with the OpenAI gpt-3.5-turbo model using item titles and descriptions as input, and then encoding the resulting profile

with text-embedding-ada002. We also provide the encoded semantic embeddings in our reproduction code. We apply RQ-Kmeans to these semantic embeddings to initialize SIDs. We set different codebook sizes according to the dataset scale, while ensuring that the total number of embeddings across all codebooks is smaller than the number of items in the dataset.

It is worth noting that many personalized federated learning works construct the recall set for validation and testing with one positive item and 99 negative items. This setting can lead to unusually high performance in some cases. In our experiments, we follow the standard recommendation setup [8] and evaluate against all items as the recall set. Therefore, our results are not directly comparable to those works.

We implement RQFedRec using PyTorch on an NVIDIA Tesla T4 GPU. We use MF as our backbone model. For fair comparison, we follow the same parameter ranges as RQFedRec and the corresponding baselines. For hyperparameter tuning, we search learning rates in $\{0.1, 0.01, 0.001\}$, weight decay in $\{0, 10^{-6}, 10^{-3}\}$, batch sizes in $\{256, 1024, 4096\}$, and SID update periods τ in $\{5, 10, 20\}$. We use Adam for local training, and detailed settings are provided in our reproduction code.

6.2 Method Comparison (RQ1)

We compare RQFedRec with all federated baselines, as shown in Table 2. We also include methods without federated aggregation, and draw the following observations.

1) Federated learning methods outperform those without aggregation, showing that sharing information across clients effectively improves recommendation performance. 2) Some personalized baselines are optimized for single user clients, so their user-level personalization advantages may weaken under our general setting. In contrast, RQFedRec focuses on improving communication and thus remains effective in this broader setting. 3) RQFedRec-R achieves

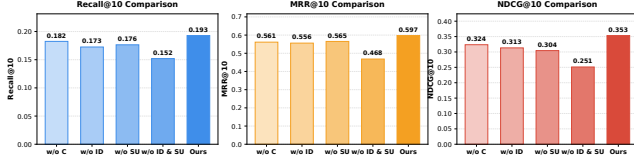


Figure 3: The ablation study of RQFedRec on the ML-100K dataset.

strong performance compared with existing federated recommendation baselines. This shows that the improvement of RQFedRec does not come from semantic priors, but mainly benefits from feature-indexed communication, information distillation, and self-refining SID update. 4) RQFedRec-S outperforms RQFedRec-R and other federated recommendation baselines on ML-100K, ML-1M, and Steam, showing that RQFedRec can effectively leverage reliable semantic information. However, when semantic information is noisy or weakly aligned with user behavior, RQFedRec-R can perform better, as observed on Movies and VideoGames.

In summary, our method generally outperforms existing baselines without any additional semantic information, while providing a reasonable scheme for utilizing additional semantic information.

6.3 Ablation Study (RQ2)

We conduct ablation studies to evaluate the contribution of the key designs in RQFedRec. Specifically, we compare RQFedRec with the following variants: 1) **w/o C**: removing feature-indexed codebook communication and using the original ID-indexed item embedding communication. 2) **w/o ID (Information Distillation)**: removing the information distillation module and directly optimizing codebooks with the recommendation loss. 3) **w/o SU (SID Update)**: removing the self-refining SID update module and keeping the initial SIDs fixed during training. 4) **w/o ID & SU**: removing both information distillation and self-refining SID update. 5) **RQFedRec**: the full model with both modules. We conduct experiments on ML-100K, and the results are shown in Figure 3.

From the results, we observe that the full RQFedRec consistently outperforms all ablated variants. 1) **w/o C** performs worse than RQFedRec on all metrics, confirming that feature-indexed codebook communication is more effective than directly communicating ID-indexed item embeddings. 2) removing either information distillation or self-refining SID update leads to clear performance degradation. The performance drop of **w/o ID** indicates that directly optimizing codebooks from sparse local interactions is insufficient, and locally learned item embeddings provide useful guidance for transferring collaborative information into codebooks. The performance drop of **w/o SU** shows that fixed SIDs cannot fully match the collaborative information that evolves during federated training. Finally, **w/o ID & SU** further degrades performance, suggesting that the two modules are complementary. These results verify the necessity of feature-indexed communication and demonstrate that both proposed modules are important for constructing effective and adaptive communication in RQFedRec.

Table 3: Comparison of download communication cost across different methods. Percentage denotes the communication cost relative to the baseline.

Dataset	Method	Parameter	Cost	Percentage
ML-100K	FedMF	$d = 128, n_i = 1682$	215296	100%
	RQFedRec	$d = 128, M = 512, L = 3$	201654	93%
ML-1M	FedMF	$d = 128, n_i = 3706$	474368	100%
	RQFedRec	$d = 128, M = 1024, L = 3$	404334	85%
Steam	FedMF	$d = 128, n_i = 5237$	670336	100%
	RQFedRec	$d = 128, M = 1024, L = 3$	408927	61%
Movies	FedMF	$d = 128, n_i = 20000$	2560000	100%
	RQFedRec	$d = 128, M = 4096, L = 3$	1632864	64%
VideoGames	FedMF	$d = 128, n_i = 20000$	2560000	100%
	RQFedRec	$d = 128, M = 4096, L = 3$	1632864	64%

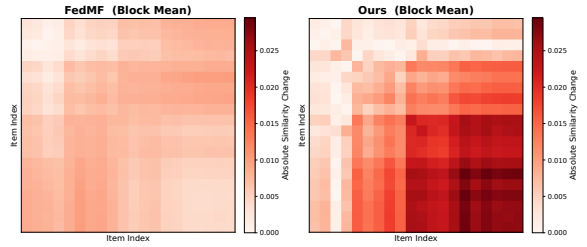


Figure 4: Visualization of item-item collaborative information updates after communication. Darker colors indicate larger information.

6.4 Analysis of Feature-Indexed Communication (RQ3)

In this section, we mainly analyze whether our method addresses the problems we identified.

6.4.1 Communication Cost. This section analyzes the communication cost of our method, including the theoretical communication cost under different paradigms, and reports the specific settings and communication cost of our method. Since downloading incurs the larger communication volume in RQFedRec, we report its download cost in Table 3. RQFedRec reduces communication on all datasets while achieving the performance reported in Table 2; further compression is possible when efficiency is prioritized over accuracy.

6.4.2 Collaborative Information Communication. We further analyze whether feature-indexed communication can transmit stronger collaborative information than ID-indexed communication. We compare RQFedRec with FedMF on ML-100K by averaging the absolute changes in item-item cosine similarity over 10 communication rounds. For visualization, we divide items into 20 consecutive index blocks; each heatmap cell shows the average similarity change between two blocks, with both methods using the same color scale.

Figure 4 shows darker regions for RQFedRec in most item-block pairs, indicating that shared codebooks transmit stronger collaborative information and enable larger changes in item relationships across rounds.

Table 4: Comparison of different noise ratios on the ML-100K dataset.

Noise ratio	Method	Recall@10	MRR@10	NDCG@10
0.05	FedMF	0.1641	0.5302	0.2951
	RQFedRec	0.1832	0.5635	0.3330
0.10	FedMF	0.1561	0.4959	0.2747
	RQFedRec	0.1715	0.5338	0.3073
0.15	FedMF	0.1341	0.4588	0.2444
	RQFedRec	0.1495	0.4855	0.2712

Table 5: Runtime comparison on ML-100K.

Method	Avg./Round	Train	Upload	Refresh	Download	vs. FedMF
FedMF	7.09s	6.93s	0.14s	0.00s	0.01s	1.00×
RQFedRec	15.81s	14.29s	0.23s	1.00s	0.29s	2.23×
FedCIA	20.09s	6.82s	0.19s	0.00s	13.08s	2.83×

6.4.3 Sensitivity to Noise. In this section, we analyze the sensitivity of our method to noise. Specifically, we simulate user misclicks by injecting noisy interactions into the raw data with different noise ratios, and then train the models on these noisy clicks. We compare our method with the traditional federated learning baseline FedMF on ML-100K, and the results are reported in Table 4.

As shown in the table, our method achieves better performance under all settings, indicating strong robustness. Moreover, as the noise ratio increases, the performance drop of our method is less pronounced, further demonstrating its resistance to noisy data.

6.4.4 Runtime Analysis. We further report the runtime cost of different methods in Table 5. Compared with FedMF, RQFedRec introduces additional computation for codebook learning and SID/codebook refresh. Therefore, the average time per communication round increases. However, the runtime cost of RQFedRec remains practical. Although RQFedRec is slower than the most basic FedMF baseline, it is still faster than some recent federated recommendation methods such as FedCIA. This shows that the additional cost of RQFedRec is moderate and acceptable, especially considering its improved recommendation performance and reduced communication cost.

6.5 Analysis of Feature-Indexed Codebook Construction (RQ4)

We examine whether the improvement of RQFedRec merely comes from external semantics or a compact codebook by comparing it with LLM-based federated methods and alternative codebook construction methods.

6.5.1 Semantic Information Utilization Analysis. Since RQFedRec can optionally use public semantic information for SID initialization, we examine whether its improvement simply comes from LLM-based item information. Table 6 compares RQFedRec with federated adaptations of RLM_con and RLM_gen [30], and AlphaFuse [11]. LLM-based semantic representations may be entangled [52] or misaligned with collaborative signals, and therefore may not consistently improve federated recommendation. In contrast, RQFedRec uses semantics only for SID initialization and subsequently refines

Table 6: Comparison with LLM-based federated recommendation methods on ML-100K.

Method	Recall@10	MRR@10	NDCG@10
FedRLM_con	0.1772	0.5775	0.3364
FedRLM_gen	0.1890	0.5622	0.3308
FedAlphaFuse	0.1836	0.5840	0.3484
RQFedRec	0.1973	0.6168	0.3650

Table 7: Comparison with different codebook construction methods on ML-100K.

Method	Recall@10	MRR@10	NDCG@10
Hash-based codebook	0.0324	0.1250	0.0475
PQ-based codebook	0.1811	0.5735	0.3239
RQ-VAE codebook	0.1901	0.6085	0.3554
RQFedRec	0.1973	0.6168	0.3650

its codebooks and SIDs, showing that its gains primarily arise from dynamic feature-indexed communication.

6.5.2 Codebook Construction Analysis. We further compare RQFedRec with other codebooks, including hash-based, PQ-based, and RQ-VAE codebooks. The results are reported in Table 7. Hashing ignores relationships among item embeddings, while PQ and RQ-VAE cannot adapt to collaborative structures that evolve during federated training. In contrast, RQFedRec distills codebooks and dynamically refines SIDs, demonstrating that effective feature-indexed communication requires both compact representation and adaptation to evolving collaborative signals.

7 Conclusion

In this paper, we proposed RQFedRec, an RQ-based federated recommendation framework that shifts conventional item embedding communication from an ID-indexed paradigm to a feature-indexed codebook paradigm. To adapt feature-indexed communication to federated recommendation, RQFedRec introduces information distillation to enable globally aligned aggregation and a self-refining SID update to capture evolving collaborative information. Extensive experiments show that RQFedRec achieves higher recommendation accuracy and lower communication cost without relying on semantic priors, while still benefiting from public semantics when available.

Acknowledgments

This work is supported by National Natural Science Foundation of China (NSFC) under the Grant No. 62372113, and Major Project of the National Social Science Fund of China (NSSFC) under the Grant No. 25&ZD260. Peng Zhang & Tun Lu are with the College of Computer Science and Artificial Intelligence, Fudan University. Tun Lu is also affiliated to the Silver-X MOE Philosophy & Social Sciences Laboratory, Fudan Institute on Aging, and Shanghai Key Laboratory of Data Science, Fudan University.

GenAI Usage Disclosure

This work uses OpenAI gpt-3.5-turbo to generate structured item feature profiles from public item titles and descriptions, and text-embedding-ada-002 to encode these profiles for semantic SID initialization, as described in the experimental settings. Generative AI tools were also used for language polishing of author-written text. They were not used to generate experimental results or scientific conclusions. All method design, experiments, analyses, and final content were conducted, reviewed, and verified by the authors.

References

- [1] Alham Fikri Aji and Kenneth Heafield. 2017. Sparse communication for distributed gradient descent. In *Proceedings of the 2017 conference on empirical methods in natural language processing*. 440–445.
- [2] Di Chai, Leye Wang, Kai Chen, and Qiang Yang. 2020. Secure federated matrix factorization. *IEEE Intelligent Systems* 36, 5 (2020), 11–20.
- [3] Yongjian Chen, Tao Guan, and Cheng Wang. 2010. Approximate nearest neighbor search by residual vector quantization. *Sensors* 10, 12 (2010), 11259–11273.
- [4] Qiwen Gu, Xuhao Zhao, Yanmin Zhu, Wenze Ma, Jiadi Yu, and Feilong Tang. 2026. Inter-attribute Semantic Correlation-Guided Federated Recommender System Against Attribute Inference Attacks. In *International Conference on Database Systems for Advanced Applications*. Springer, 121–137.
- [5] Shengkang Gu, Jiahao Liu, Dongsheng Li, Guangping Zhang, Mingzhe Han, Hansu Gu, Peng Zhang, Ning Gu, Li Shang, and Tun Lu. 2026. LLM Agent-based Shilling Attack on Recommender Systems. In *Proceedings of the Nineteenth ACM International Conference on Web Search and Data Mining*. 163–173.
- [6] 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*. 1687–1696.
- [7] 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.
- [8] 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.
- [9] 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.
- [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] Guoqing Hu, An Zhang, Shuo Liu, Zhibo Cai, Xun Yang, and Xiang Wang. 2025. Alphafuse: Learn id embeddings for sequential recommendation in null space of language embeddings. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1614–1623.
- [12] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)* 20, 4 (2002), 422–446.
- [13] Fan Lai, Xiangfeng Zhu, Harsha V Madhyastha, and Mosharaf Chowdhury. 2021. Oort: Efficient federated learning via guided participant selection. In *15th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 21)*. 19–35.
- [14] Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. 2022. Autoregressive image generation using residual quantization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 11523–11532.
- [15] 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.
- [16] 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.
- [17] Z Li, G Long, and T Zhou. 2024. Federated Recommendation with Additive Personalization. In *The Twelfth International Conference on Learning Representations*.
- [18] Guanyu Lin, Feng Liang, Weiye Pan, and Zhong Ming. 2020. Fedrec: Federated recommendation with explicit feedback. *IEEE Intelligent Systems* 36, 5 (2020), 21–30.
- [19] 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.
- [20] 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.
- [21] Jiahao Liu, Dongsheng Li, Hansu Gu, Peng Zhang, Tun Lu, Li Shang, and Ning Gu. 2025. Unbiased collaborative filtering with fair sampling. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2555–2559.
- [22] Jiahao Liu, Xueshuo Yan, Dongsheng Li, Guangping Zhang, Hansu Gu, Peng Zhang, Tun Lu, Li Shang, and Ning Gu. 2025. Improving LLM-powered Recommendations with Personalized Information. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2560–2565.
- [23] Siyi Liu and Yujia Zheng. 2020. Long-tail session-based recommendation. In *Proceedings of the 14th ACM conference on recommender systems*. 509–514.
- [24] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*. 43–52.
- [25] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. Pmlr, 1273–1282.
- [26] Frank McSherry and Ilya Mironov. 2009. Differentially private recommender systems: Building privacy into the netflix prize contenders. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 627–636.
- [27] Apurva Pathak, Kshitiz Gupta, and Julian McAuley. 2017. Generating and personalizing bundle recommendations on steam. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*. 1073–1076.
- [28] Vasileios Perifanis and Pavlos S Efraimidis. 2022. Federated neural collaborative filtering. *Knowledge-Based Systems* 242 (2022), 108441.
- [29] Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Tran, Jonah Samost, et al. 2023. Recommender systems with generative retrieval. *Advances in Neural Information Processing Systems* 36 (2023), 10299–10315.
- [30] Xubin Ren, Wei Wei, Lianghao Xia, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. 2024. Representation learning with large language models for recommendation. In *Proceedings of the ACM web conference 2024*. 3464–3475.
- [31] Zehua Sun, Yonghui Xu, Yong Liu, Wei He, Lanju Kong, Fangzhao Wu, Yali Jiang, and Lihzen Cui. 2024. A survey on federated recommendation systems. *IEEE Transactions on Neural Networks and Learning Systems* (2024).
- [32] 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.
- [33] Ziye Tong, Jiahao Liu, Weimin Zhang, Hongji Ruan, Derick Tang, Zhanpeng Zeng, Qinsong Zeng, Peng Zhang, Tun Lu, and Ning Gu. 2026. RQ-GMM: Residual Quantized Gaussian Mixture Model for Multimodal Semantic Discretization in CTR Prediction. *arXiv preprint arXiv:2602.12593* (2026).
- [34] Paul Voigt and Axel Von dem Bussche. 2017. The eu general data protection regulation (gdpr). *A practical guide, 1st ed., Cham: Springer International Publishing* 10, 3152676 (2017), 10–5555.
- [35] Ellen M Voorhees et al. 1999. The TREC-8 question answering track report. In *Trec*, Vol. 99. 77–82.
- [36] Jianyu Wang, Qinghua Liu, Hao Liang, Gauri Joshi, and H Vincent Poor. 2020. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Advances in neural information processing systems* 33 (2020), 7611–7623.
- [37] Wenjie Wang, Honghui Bao, Xinyu Lin, Jizhi Zhang, Yongqi Li, Fuli Feng, See-Kiong Ng, and Tat-Seng Chua. 2024. Learnable item tokenization for generative recommendation. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*. 2400–2409.
- [38] Jie Wen, Zhixia Zhang, Yang Lan, Zhihua Cui, Jianghui Cai, and Wensheng Zhang. 2023. A survey on federated learning: challenges and applications. *International journal of machine learning and cybernetics* 14, 2 (2023), 513–535.
- [39] Longtao Xiao, Haozhao Wang, Cheng Wang, Linfei Ji, Yifan Wang, Jieming Zhu, Zhenhua Dong, Rui Zhang, and Ruixuan Li. 2025. Unger: Generative recommendation with a unified code via semantic and collaborative integration. *ACM Transactions on Information Systems* 44, 2 (2025), 1–31.
- [40] 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.
- [41] Tianzi Zang, Yidan Fan, Yixin Chen, Juan Li, Tong Zhang, and Yanmin Zhu. 2025. Mutual Knowledge Distillation and Contrastive Learning between Multi-View Graphs for Cross-Domain Recommendation. *ACM Transactions on Information Systems* 44, 2 (2025), 1–31.
- [42] Tianzi Zang, Yanmin Zhu, Haobing Liu, Ruohan Zhang, and Jiadi Yu. 2022. A survey on cross-domain recommendation: taxonomies, methods, and future directions. *ACM Transactions on Information Systems* 41, 2 (2022), 1–39.
- [43] 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.
- [44] 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.
 - [45] Honglei Zhang, Haoxuan Li, Jundong Chen, Sen Cui, Kunda Yan, Abudukelimu Wuerkaixi, Xin Zhou, Zhiqi Shen, and Yidong Li. 2026. Beyond similarity: Personalized federated recommendation with composite aggregation. *ACM Transactions on Information Systems* 44, 2 (2026), 1–28.
 - [46] Honglei Zhang, Zhiwei Li, Haoxuan Li, Xin Zhou, Jie Zhang, and Yidong Li. 2026. Transfr: Transferable federated recommendation with adapter tuning on pre-trained language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 40. 28212–28220.
 - [47] Honglei Zhang, Fangyuan Luo, Jun Wu, Xiangnan He, and Yidong Li. 2023. LightFR: Lightweight federated recommendation with privacy-preserving matrix factorization. *ACM Transactions on Information Systems* 41, 4 (2023), 1–28.
 - [48] Jun Zhang, Dugang Liu, Xing Tang, Xiuqiang He, and Zhong Ming. 2026. FedMM: Federated Collaborative Signal Quantization for Multi-Market CTR Prediction. In *Proceedings of the 49th international ACM SIGIR conference on research and development in information retrieval*. 2343–2353.
 - [49] Xuhao Zhao, Zhongrui Zhang, Yanmin Zhu, Zhaobo Wang, Wenze Ma, Jiadi Yu, and Feilong Tang. 2025. Social Relation-Level Privacy Risks and Preservation in Social Recommender Systems. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1728–1737. doi:10.1145/3726302.3730086
 - [50] Xuhao Zhao, Yanmin Zhu, Chunyang Wang, Mengyuan Jing, Jiadi Yu, and Feilong Tang. 2023. Task-Difficulty-Aware Meta-Learning with Adaptive Update Strategies for User Cold-Start Recommendation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 3484–3493. doi:10.1145/3583780.3615074
 - [51] Guorui Zhou, Jiaxin Deng, Jinghao Zhang, Kuo Cai, Lejian Ren, Qiang Luo, Qianqian Wang, Qigen Hu, Rui Huang, Shiyao Wang, et al. 2025. OneRec Technical Report. *arXiv preprint arXiv:2506.13695* (2025).
 - [52] Jiankui Zhou, Jing Yao, Xiaoyuan Yi, Peng Zhang, Ning Gu, Zhan Hu, Xing Xie, and Tun Lu. 2026. Disentangling Consensus and Value-Specific Representations for Controllable Pluralistic Value Alignment of LLMs. In *Forty-third International Conference on Machine Learning*. <https://openreview.net/forum?id=0xiTAhvLj6>