FedRL: A Reinforcement Learning Federated Recommender System for Efficient Communication Using Reinforcement Selector and Hypernet Generator

YICHENG DI

School of Artificial Intelligence and Computer Science, Jiangnan University, Wuxi, Jiangsu, China, diyicheng@stu.jiangnan.edu.cn

HONGJIAN SHI

School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China, shhjwu5@sjtu.edu.cn

RUHUI MA

School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China, ruhuima@sjtu.edu.cn

HONGHAO GAO

Shanghai University, Shanghai, Shanghai, China, gaohonghao@shu.edu.cn

YUAN LIU

Jiangnan University, Wuxi, Jiangsu, China, lyuan1800@jiangnan.edu.cn

WEIYU WANG

Rutgers University, New Brunswick, New Jersey, United States, Ww367@rutgers.edu

The field of recommender systems aims to predict users' latent interests by analyzing their preferences and behaviors. However, privacy concerns about user data collection lead to challenges such as incomplete initial information and data sparsity. Federated learning has emerged to address these privacy issues in recommender systems. However, federated recommender systems face heterogeneity among edge devices regarding data features and sample sizes. Moreover, differences in computational and storage capabilities introduce communication overhead and processing delays during parameter aggregation at the third-party server. This paper introduces a framework named A Reinforcement Learning Federated Recommender System for Efficient Communication Using Reinforcement Selector and Hypernet Generator (FedRL) to address the proposed issues. The Reinforcement Selector (RLS) dynamically selects participating edge devices and helps maximize their use of local data resources. Meanwhile, the Hypernet Generator (HNG) optimizes communication bandwidth consumption during the federated learning parameter transmission, enabling rapid deployment and updates of new model architectures or hyperparameters. Furthermore, the framework incorporates item attributes as content embeddings in edge devices' recommender models, enriching them with global information. Real-world dataset experiments demonstrate that the proposed solution balances recommender quality and communication efficiency. The code for this work is publicly available on GitHub: https://github.com/diyicheng/FedRL.

CCS CONCEPTS • Computing methodologies • Artificial intelligence • Distributed artificial intelligence • Cooperation and coordination

ACM 2770-6699/2024/07-ART
http://dx.doi.org/10.1145/3682076

Additional Keywords and Phrases: Federated learning, hypernetwork, reinforcement learning, federated recommender system, attention mechanism

1 Introduction

Recommender systems become essential tools in aiding users in effectively exploring extensive data sets to extract relevant knowledge, owing to the exponential increase in information volume. A key characteristic of recommender systems is their ability to perform personalized recommendations by analyzing user preferences from data. However, to make accurate recommendations in recommender scenarios, these systems often require edge devices to permit the collection of user's private data. In real-life scenarios, privacy restrictions and incomplete user information present challenges that hinder the ability of these systems to capture user preferences effectively. Assisting users in acquiring relevant knowledge within the domain of initial information incompleteness and data sparsity is a formidable task for recommender systems. The issue of incomplete initial information can result in the system failing to comprehend the user's interests and preferences, thus impeding its ability to provide targeted recommendations. Many conventional approaches attempted to address this issue, such as analyzing users' historical profiles or item attributes [1], but these methods still require uploading sensitive private data to third-party servers. Moreover, machine learning models trained solely on historical data often suffer from shortcomings in generalization. The capacity of the model to generalize is essentially based on the quality and variety of the training data. Due to practical limitations in real-life applications, users are often reluctant to upload their private data to the system, leading to data scarcity challenges for recommender system models. This limitation not only poses difficulties in model training but also results in recommender bias. With limited interactions between new users and items in the system, recommender systems struggle to accurately learn user preferences from historical information.

In recent research, Federated Learning [2, 3, 4] emerge as a promising approach applied to address the challenges of initial information incompleteness and data sparsity in recommender systems. Federated Learning is a distributed learning method designed to tackle data privacy and security concerns. It allows knowledge sharing among multiple participants without the need to transmit raw data to a central server. In recent works, most recommender systems based on Federated Learning [5, 6] adopt model aggregation in order to generate a global model. The global model can effectively leverage data features from various participants to address the issues of initial information incompleteness and data sparsity in the recommender system. These methods show great potential when applying Federated Learning to scenarios with initial information incompleteness and data sparsity. However, in federated learning, client devices need to send updated model parameters to the central server for aggregation, which involves a large amount of communication. Due to bandwidth constraints and unstable network conditions, many methods still face high communication overhead and processing delays, and even communication failures can occur. In the VerFedGNN proposed by Mai et al. [75], they apply Graph Neural Networks (GNN) in federated recommender systems. GNN learns by propagating messages across the graph, which may result in the need to transmit a large amount of information in each round of federated learning. Moreover, each round of federated learning requires message passing and parameter updates over the entire graph, leading to higher processing delays. In ReFRS proposed by Imran et al. [76], they utilize a variational autoencoder to

capture users' temporal preferences among a series of interacting items. The variational autoencoder captures distribution features of the data through latent variables, requiring more communication resources when transmitting latent representations. Particularly, when dealing with a large number of edge devices, the substantial data interactions between edge devices and the server can lead to significant communication costs. In order to update the global model, the third-party server must gather parameters from all devices. However, the significant communication overhead and processing delays are inevitable due to the restricted computational capability and bandwidth of edge devices.

In this context, reinforcement learning, due to its ability to adjust the device selection strategy based on real-time performance and environmental changes, can help the model reduce unnecessary communication and data transmission. Furthermore, reinforcement learning can learn shared policies in Federated Learning, allowing the global model's decisions and knowledge to be transferred to edge devices, thus reducing communication overhead. The proposed approach aligns well with the Federated Learning framework. By employing reinforcement learning to learn and optimize resource allocation strategies, it can efficiently distribute computing and communication resources to critical edge devices, thereby enhancing the overall efficiency and responsiveness of Federated Learning. Particularly, the decision-making process for edge device selection in Federated Learning is intricate, where reinforcement learning continuously learns and refines selection strategies through interactions with the environment, progressively improving the effectiveness and performance of Federated Learning procedure.

Recent research [7, 8, 9] show that Hypernetworks [10] are an effective strategy for reducing communication overhead and delays. Hypernetworks are deployed on the third-party server and serve the primary function of generating and controlling the weights and parameters of other neural networks. By allowing Hypernetworks to generate and adjust the weights and parameters of subnetworks on edge devices, the need for directly transmitting large amounts of parameters to the third-party server is eliminated. This parameter generation approach significantly reduces the demand for network communication, thereby lowering communication overhead. Moreover, as Hypernetworks can generate subnetwork parameters locally on the devices, the frequent interactions with the third-party server are avoided, resulting in reduced communication delays. Nowadays, these technologies are applied in the context of Federated Learning, and one common application is to utilize the learned initial parameters of the global model to enable local devices to adapt to new tasks and data more swiftly, thus reducing communication overhead and delays. Segev et al. [73] employ hypernetworks to reduce the communication costs of federated learning in medical image tasks. Additionally, Hypernetworks can dynamically generate weights and parameters of the network based on input data, allowing the network to adapt to different input instances. Zhang et al. [74] utilize hypernetworks to generate personalized local models for edge devices, aiming to implement tailored solutions in the context of the Internet of Things. This flexibility enhances the personalization and adaptability of the network, thereby improving its performance across diverse tasks and data. Hypernetwork technology proved to be an effective tool for reducing communication overhead and lowering communication delays in various Federated Learning tasks. Many Federated Learning methods that utilize Hypernetworks [11] demonstrat significant performance improvements, especially in reducing communication overhead and delays.

For the purpose of addressing the challenges of high communication overhead and delays in existing Federated Recommender Systems, we introduce two key components: the RLS related to Reinforcement Learning techniques and the HNG associated with Hypernetwork technology. The incorporation of these components is motivated by two primary factors. Firstly, the RLS utilizes Reinforcement Learning to effectively balance resource allocation based on the computational capacity and data quality of individual edge devices. By avoiding excessive workload on specific devices, this approach efficiently mitigates communication overhead and delays, leading to improved performance and overall system efficiency. Secondly, the HNG enables dynamic generation of model parameters tailored to the data features and requirements of different edge devices. This adaptive approach allows each edge device to receive customized model updates, thereby reducing unnecessary communication and streamlining the process. For the purpose of effectively tackling the aforementioned issue, we put forword an unique FedRL framework: A Reinforcement Learning Federated Recommender System for Efficient Communication Using Reinforcement Selector and Hypernet Generator. The novelty of the FedRL lies in several key aspects. Firstly, it deploys the HNG on a third-party server to generate network parameters for the recommender networks on edge devices. Secondly, it incorporates the RLS to actively choose the optimal edge device during each round of communication in federated learning. Additionally, it utilizes content embeddings of item attributes as part of the global information. Finally, it employs attention residual blocks to focus on attribute information that significantly impacts recommender effectiveness. The efficacy of FedRL has been demonstrated by comprehensive experimentation conducted on four datasets, Movielen1M, Ciao, Douban-Movie, and Amazon Movies. In comparison to other baselines, FedRL exhibits respective improvement of 5.1%, 7.7%, 6.8% and 6.7% across four datasets. In the ablation experiments, FedRL further validates the essentiality of each component designed within the Federated Learning framework. The key contributions of this study can be compactly summarized as follows:

(1) We present an innovative and efficient communication Reinforcement Learning Federated Recommender System framework that reduces communication overhead, lowers communication delays, and quickly adapts to new users.
(2) The integration of RLS and HNG contributes to effectively reducing communication costs, improving communication efficiency, and achieving satisfactory performance in scenarios with initial information incompleteness and data sparsity.
(3) We perform thorough experiments on three actual datasets. The results of these experiments demonstrate the efficacy of our FedRL framework in enhancing communication efficiency in scenarios with initial information incompleteness and data sparsity while maintaining competitive recommender performance.

The subsequent sections of this paper are structured in the ensuing manner: Section 2 elucidates the pertinent research within this domain. Following this, Section 3 delves into foundational insights, encompassing definitions and formulations. Proceeding chronologically, Section 4 furnishes an in-depth elaboration of the FedRL model. The outcomes of the experiments are deliberated upon in Section 5. Lastly, Section 6 culminates with a summarization of the paper's content, coupled with a discourse on potential avenues for future exploration.

2 Related Work

2.1 Federated Learning for Recommender System

The concept of Federated Learning is initially proposed by McMahan et al. [12] from Google. This approach presents a privacy-preserving method to distribute sensitive user data across edge devices, allowing for learning a shared model through aggregating local computation updates. The primary objective of Federated Learning is to achieve data privacy protection and model sharing by performing localized model training and inference on distributed devices [13, 14, 15]. Federated Learning has emerged as a promising machine learning approach suitable for scenarios involving large-scale data and sensitive information. In contrast to traditional methods that require centralizing the dataset on a single third-party server or cloud platform for training, Federated Learning mitigates privacy risks by avoiding the transmission of users' sensitive data to a central server for processing. In recent research [16, 17, 18], the application of Federated Learning in recommender systems has gained attention as a means to protect user privacy data while ensuring robust recommender performance. Ammad-Ud-Din et al. [19] introduce the first federated recommender algorithm, FCF, which effectively addresses the issue of potential privacy leakage when computing item feature vectors. They propose a mechanism to retain user's implicit feedback data locally and upload only the gradients of item feature vectors to a third-party server for updating, thereby preserving user privacy while maintaining accurate recommendations. Lin et al. [20] point out that sending model updates to third-party servers might inadvertently reveal sensitive data, resulting in privacy leakage. To address this concern, they propose a solution involving user averages and mixed padding to assign virtual ratings to unrated items, effectively preventing the disclosure of explicit data. In a related study, Liang et al. [21] present a lossless federated recommender system with explicit feedback. They observe that assigning virtual ratings to randomly sampled items introduce noise during the model training process, potentially impacting recommender performance. To overcome this challenge, the researchers propose a method that strategically assigns denoising edge devices in a privacy-aware manner to eliminate noise, leading to improved recommender accuracy. Considering the personalized nature of user preferences in federated recommender systems, Di et al. [22] introduce the use of meta-learning methods to capture individualized user preferences effectively. To address the item cold start problem in recommender systems, Wahab et al. [23] introduce a powerful federated recommender system. This system utilize a DDQN intelligent scheduling strategy, which rely on trust scores to select recommenders for prediction. Tan et al. [24] develop an online federated recommender system platform, integrating several prominent recommender algorithms. For instance, FM [25], which amalgamates the concept of matrix factorization and feature interaction to address predictive issues within recommender systems. Wide & Deep [26], simultaneously combining linear models and deep neural network models to handle extensive features within large-scale data. Matrix factorization [27], decomposing rating matrices into multiple lower-dimensional dense matrices to extract latent features. These commonly used recommender algorithms were merged with federated learning methods. Jie et al. [80] propose a federated recommender algorithm based on historical parameter clustering. Clients use a time decay factor to weighingly average historical learning parameters and the

global parameters sent by the server. They utilize user embedding features for personalized recommender on the server side. Sun et al. [83] further formulate the recommender problem as minimizing the long-term average system cost in a Markov decision process to enhance user experience quality. The collective findings from these studies demonstrate the efficacy of integrating federated learning techniques into recommender systems.

However, the aforementioned conventional federated recommender systems typically require communication among participants in each iteration, which leads to high communication costs and diminishes the effectiveness of recommendations. Yuan et al. [81] eliminate user contributions by rolling back and calibrating historical parameter updates. They introduce a small negative sampling method to reduce the number of item embedding updates to improve operational efficiency. Qu et al. [82] employ a semi-decentralized federated learning approach to reduce communication costs. They utilize predicted interaction nodes to connect independent ego graphs to enhance scalability. Hu et al. [70] utilize graph convolutional networks to process high-order connectivity information. However, in each iteration, a substantial amount of graph data, comprising node features and adjacency matrices, may need to be transmitted between edge devices and third-party servers, resulting in significant communication overhead. Vyas et al. [71] adapt federated recommender methods to the domain of driving behavior analysis. Nevertheless, in this scenario, diverse edge devices possess varying bandwidths, where devices with lower bandwidths require more time to complete the transmission of model parameters, thereby augmenting the overall communication overhead and latency. Li et al. [72] employ heterogeneous graph techniques for cross-domain federated recommendations. Nevertheless, within heterogeneous graphs, updating model parameters necessitates consideration of complex relationships among different types of nodes and edges, leading to increased communication overhead. Given that participants are distributed across different devices or locations, the frequent communication in traditional federated recommender system approaches increases the time cost of communication, resulting in higher latency.

2.2 Optimizing Federated Learning through Reinforcement Learning

The rapid development of reinforcement learning has led to the emergence of federated reinforcement learning methods [41, 42, 43]. In federated learning, edge devices may have different computing resources and capabilities. Reinforcement learning can optimize resource allocation during the edge device selection process. By considering factors such as computing resources and network bandwidth of edge devices, reinforcement learning can formulate reasonable strategies for task assignment to suitable edge devices to achieve optimal training performance. Recent works [44, 45, 46] have further developed federated reinforcement learning systems to optimize resource allocation and improve communication efficiency while maintaining edge devices' personalization. Li et al. [47] argue that traditional methods cannot achieve long-term optimal performance, and they propose using a multi-agent reinforcement learning framework for collaborative computing to achieve long-term optimal performance. This approach allows agents to collaboratively explore the environment, leading to faster convergence and reduced communication latency. Wang et al. [48] suggest enhancing the training efficiency of the Q-learning network while accounting for constrained computing and storage capabilities.

Additionally, an attention mechanism is employed during the weight aggregation phase to prevent discrepancies in model quality across edge devices. Yu et al. [49] introduce fast and slow time-scale learning processes in federated reinforcement learning to achieve real-time and low-cost computation. They jointly optimize resource allocation and computation offloading to minimize total latency. Xu et al. [50] discover that the convergence speed of single-agent reinforcement learning is affected by the heterogeneity of edge devices. To reduce the policy evaluation error caused by interactions between multiple devices, they used a multi-agent reward mechanism to capture the smoothness approximation in federated learning when employing heterogeneous edge devices. FAVOR [51] utilize deep Q-learning to select a subset of edge devices in each communication round, aiming to maximize global rewards to counteract biases introduced by non-iid data. This approach improve model accuracy and reduced communication rounds. Xie et al. [84] have devise a KL divergence penalty method for gradient-based policy reinforcement learning federated systems to expedite the algorithm's training process. Tiwari et al. [85] utilize a multi-agent federated reinforcement learning strategy combined with dynamic environmental changes to enhance the operational efficiency of devices in IoT environments. Zhou et al. [86] employ a dual-layer reinforcement learning approach and collaborative strategies to assist in client node selection and global frequency optimization during the federated learning process. Zhang et al. [52] present the latest work combining reinforcement learning with federated learning. They leverage device performance statistics and training behavior to perform optimal edge device selection based on optimization objectives. This work is highly relevant to our study.

However, due to the potential uncertainty in the dynamics of the environment and reward signals in reinforcement learning, especially in the context of federated learning, user behavior and feedback can be influenced by multiple factors. The aforementioned methods that utilize reinforcement learning to optimize the federated process may introduce biases or instability in the decision-making process, thereby affecting the effectiveness of recommendations.

2.3 Hypernetworks

Hypernetworks aim to reduce communication overhead and enhance model performance by utilizing a deep neural network to output the weights of another target network [28, 29, 30]. They have found widespread applications in various machine learning scenarios, such as 3D content generation [31], computer vision [32], multi-objective optimization [33], language modeling [34], and building blocks [35], among others. Due to their ability to facilitate knowledge sharing while reducing data transmission during communication, hypernetworks are highly suitable for integration with federated learning. Litany et al. [36] employ hypernetworks as knowledge aggregators in federated learning to accommodate heterogeneous edge device architectures. They use the description of edge device architectures as inputs to the hypernetwork, which then generated the weights to fill the edge device architectures. Their approach exhibite significant advantages in handling unknown architectures, improving model accuracy, and notably reducing the time required for the model to converge. Zhao et al. [37] utilize hypernetworks for soft weight sharing. They performe meta-learning on the hypernetwork and adjusted its

embeddings to create few-shot learners, which significantly compress the parameter count to optimize communication. Shamsian et al. [38] employ hypernetworks to collaboratively train personalized models within a federated architecture, considering the diversity of edge device data and reducing communication costs. They utilize hypernetworks on the third-party server to generate models for each edge device. This architecture achieves parameter sharing across edge devices while generating diverse personalized models for each edge device. As the parameters of the hypernetwork do not directly participate in communication, this approach decouples communication costs from the size of trainable models, effectively reducing communication overhead. Galanti et al. [39] conduct a comparative study between hypernetworks and embedding-based methods, highlighting the modularity of hypernetworks. They found that hypernetworks have significantly fewer trainable parameters compared to standard neural networks and embedding methods. This demonstrates the potential of hypernetworks in reducing the amounts of transmitted parameters during communication. Chen et al. [87] employ a hypernetwork to generate the parameters of a re-ranking model based on different preference weights, facilitating online adjustment of preference weights. Shen et al. [88] incorporate temporal features as input to the hypernetwork and dynamically adjust the recommendation model to adapt to time-varying user preferences. Liu et al. [89] utilize an efficiently parameterized hypernetwork to extract user interests for each task from various types of behavioral sequences, mitigating interference between tasks. Ma et al. [40] combines hypernetworks with federated learning. They address the issue of delayed model convergence caused by the weighted aggregation method used for generating personalized models. By utilizing hypernetworks on the third-party server to assess the mutual contribution of edge devices at a higher granularity level, they achieve better personalized models for each edge device while reducing communication overhead during training. This work is highly relevant to our research.

However, pFedLA does not address the bias introduced by non-iid data on each edge device, nor does it incorporate dynamic selection of participating edge devices for communication in each round. This can potentially result in certain devices being overlooked or participating less in the communication during the training process, thereby impacting the recommender performance. In our approach, we adopt hypernetworks as the backbone network on the third-party server, aiming to address the aforementioned limitations.

3 Preliminary

Within this section, we offer an introduction to the core context and establish key terminologies. A concise overview of the primary notations employed within this study is provided in Table 1.

3.1 Definitions

The objective of FedRL is to collaboratively train models to predict new users' ratings for items in scenarios with incomplete initial information and sparse data, while ensuring efficient communication. Let $U = \{u_j^{(1)}, u_j^{(2)}, \ldots, u_j^{(N)}\}$ and $I = \{i_j^{(1)}, i_j^{(2)}, \ldots, i_j^{(N)}\}$ represent the user and item sets, respectively. The number of edge devices is denoted by N.

Table 1. Main notions.

Symbol	Description
$U; I;$ R	user set; item set; score set
D_i	data on the *i*-th client
$K; k_i$	size of all client datasets; size of the dataset on the *i*-th client
$p_{u_c}; q_{i_c}; p_{\hat{u}_c}$	user embeddings; item embeddings; user preference embeddings
$q_{i_c}^j$	content embedding of item properties
$u_j^{(i)}, i_j^{(i)}, R_j^{(i)}$	specific user; specific item; specific rating
θ_i	recommender model parameters on the *i*-th client
$A(i); H(i)$	item attribute representation; one-hot coding of item attribute content
$R_{u_c,i_c}; \hat{R}_{u_c,i_c}$	actual scoring of items; predicted scoring of items
$G_{t,i}^s; G_{y,i}^c$	local training time of client i at round t; communication delay of client i uploading to the server at round t
G_t	processing delay for round t
C_i^t	communication cost of updates incurred by client i to the third-party server in round t
$a_1; a_2; a_3$	coefficient of the importance of the control target
l_i^t	test loss of agent i in round t
$h_{t,i}^d$	historical test training delay
$\ell_i; L_i$	loss function of client i; average loss of the *i*-th client training

Provided with the set of ratings $\mathrm{R} = \{R_j^{(1)}, R_j^{(2)}, \ldots, R_j^{(N)}\}$, the objective is to fulfill user ratings for various items. Within the framework of federated learning, individual user data is maintained locally on edge devices, characterized by the subsequent description:

Definition 1(Client). An edge device, designated as the "client," is tasked with the storage of data pertaining to user-item rating information. Each distinct client is uniquely linked with an individual user, referred to herein as u. Furthermore, the dataset housed within the *i*-th client is specifically denoted as D_i.

Definition 2(Server). The server, referred to as a third-party entity, is tasked with overseeing multiple clients and orchestrating the federated training procedure. Its role encompasses receiving solely essential data for model enhancements, without engaging in the exchange of raw data preserved on the clients.

In this study, we operate under the assumption that clients are capable of exclusively providing precise data and are unable to interfere with the training regimen. The primary objective of FedRL is to collectively train client models to forecast ratings for new users, leveraging incomplete and fragmentary rating information. The definition of FedRL is as follows:

Definition 3(FedRL). In the context of N individual clients and a central server, FedRL can efficiently predict unknown ratings in each client i based on the given partially known data $D_i = \{(u_j^{(i)}, i_j^{(i)}, R_j^{(i)})\}_{i=1}^{k_i}, (1 \le i \le N)$ on each client without accessing the raw data. This approach ensures data privacy and security while maintaining effective

communication during the federated learning process.

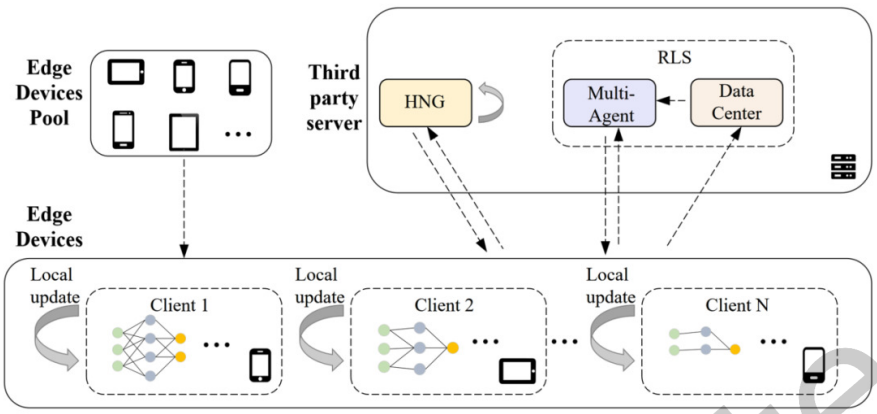

Figure 1. Illustration of a reinforcement learning federated recommender system framework for efficient communication (FedRL). In each client, we use the HNG to generate the parameters on the client's recommender network. Then, the RLS selects the eligible clients to participate in the aggregation.

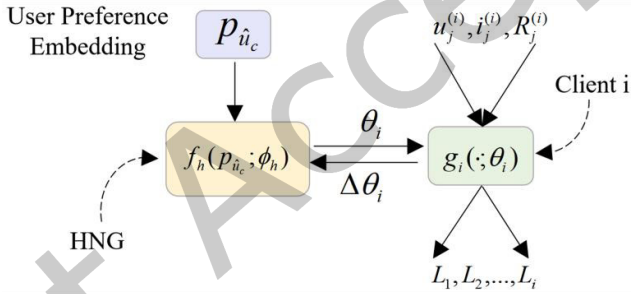

Figure 2. The framework of the HNG. The HNG, represented as $f_h(\cdot;\phi_h)$, is deployed on the server and takes user preference embeddings $p_{\hat{u}_c}$ as input to generate the parameters θ_i for each client i's recommender model. Client i undergoes multiple local optimization steps and returns the update direction $\Delta\theta_i$ for their individual model.

3.2 Problem Formulation

In FedRL, the objective is to facilitate direct collaboration among multiple clients to collectively train a model for predicting ratings of new users while maintaining efficient communication. Consider a server with N individual clients, where $D_i = \{(u_j^{(i)}, i_j^{(i)}, R_j^{(i)})\}_{i=1}^{k_i}, (1 \le i \le N)$ represents the known data on the i-th client, with $u_j^{(i)}$ and $i_j^{(i)}$ being the j-th input user and item, respectively, and $R_j^{(i)}$ denoting the corresponding rating. The dataset size on the i-th client is denoted by k_i. Let θ_i represent the parameters of the recommender network on the client. The objective of FedRL can be formulated as follows:

$$\Phi^* = arg\ \min_{\Phi} \sum_{i=1}^{N} \frac{k_i}{K} L_i(\theta_i) \tag{1}$$

In this context, $L_i(\theta_i) = \frac{1}{k_i} \sum_{j=1}^{k_i} \ell_i(u_j^{(i)}, i_j^{(i)}, R_j^{(i)}; \theta_i)$ represents the loss function trained on the *i*-th client, and $\Phi = \{\theta_1, \theta_2, \ldots, \theta_N\}$ represents the collection of parameters of the recommender network across all clients.

4 Proposed Framework

In this section, we introduce our proposed FedRL, comprising two key components: Reinforcement Selector and Hypernetwork Generator. The proposed framework is illustrated in Figure 1.

4.1 HyperNet Generator

The HNG is a crucial component in FedRL responsible for generating the parameters of the recommender networks on each client and facilitating collaborative model training. It is designed as a deep neural network that generates the recommender network's parameters based on the user's preference embedding. The HNG simultaneously learns a set of recommender networks. Specifically, the HNG can be formulated as follows:

$$f_h(\cdot; \phi_h) \tag{2}$$

Where $f_h(\cdot)$ is parameterized by ϕ_h and represents the HNG. The HNG is responsible for generating parameters for the recommender networks on each client. The recommender network can be formulated as follows:

$$g_i(\cdot; \theta_i) \tag{3}$$

Where $g_i(\cdot)$ is parameterized by θ_i and represents the recommender network. Each client i has a corresponding recommender network $g_i(\cdot)$. The HNG is deployed on the server and takes user preference embeddings $p_{\hat{u}_c}$ as input, as shown in Figure 2. Given user preference embeddings $p_{\hat{u}_c}$, the HNG outputs parameters θ_i for the *i*-th client. The client performs multiple local optimization steps to obtain $\tilde{\theta}_i$ and returns an update direction $\Delta\theta_i = \tilde{\theta}_i - \theta_i$. The HNG achieves information sharing among clients through shared parameters ϕ_h.

Based on the above setup, we can adjust the objective of FedRL to:

$$\Phi^* = arg\ \min_{p_{\hat{u}_1}, \ldots, p_{\hat{u}_N}, \phi_h} \sum_{i=1}^{N} \frac{k_i}{K} L_i(f_h(p_{\hat{u}_c}; \phi_h)) \tag{4}$$

Shamsian et al. [38] employ the chain rule to reduce the number of parameters transmitted during the federated learning process. We find that the chain rule is similarly applicable in our approach. we can use the chain rule to find $\nabla_{\phi_h} L_i = (\nabla_{\phi_h} \theta_i)^T \nabla_{\theta_i} L_i$, so when the clients return the parameters to the HNG, they only need to transmit $\nabla_{\theta_i} L_i$. This

allows us to further use a more general update rule $\Delta\phi_h = (\nabla_{\phi_h}\theta_i)^T \Delta\theta_i$. Therefore, HNG helps decouple the relationship between the hypernetwork size and communication costs in FedRL. The number of data exchanged among clients and server is determined by the size of the recommender networks on the clients, i.e., the parameters ϕ_h of the HNG are updated by

$(\nabla_{\phi_h}\theta_i)^T(\tilde{\theta}_i - \theta_i)$. This optimization aids in the efficient utilization of communication resources, leading to improved model performance and efficiency.

4.2 Recommender Network

The recommender networks are deployed on the clients, and each client i has its corresponding recommender network denoted as $g_i(\cdot;\theta_i)$. The user preference embedding $p_{\hat{u}_c}$ is composed of a mixture of user preference bases, and its formula is as follows:

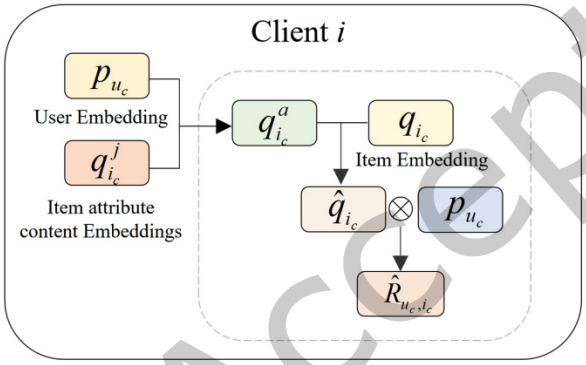

Figure 3. The overall structure of the recommender network on the client. The recommender network utilizes user embeddings p_{u_c} and item embeddings q_{i_c} to predict user preferences for items. The content embeddings of the items $q^j_{i_c}$ are incorporated as global information into the recommender network.

$$p_{\hat{u}_c} = B_{u_c} p_{u_c^*} \tag{5}$$

The user interest bases are the column vectors in matrix B_{u_c} represented as b_{u_c}. $p_{u_c^*}$ is formed by user IDs and stores the weights of user preference bases. By modifying the information in $p_{u_c^*}$, we can efficiently optimize the user preference embedding for new users. Since all the parameters in the recommender network are generated by HNG with $p_{\hat{u}_c}$ as input, we can simply adjust $p_{u_c^*}$ to quickly adapt $p_{\hat{u}_c}$ and ensure that the parameters of the recommender network on the client are appropriately generated.

Each item can be described by a set of attributes represented as $A(i) = (A_1^{(i)}, A_2^{(i)}, \ldots, A_N^{(i)})$. The content of the attributes can be encoded as one-hot vectors $H(i) = (H_1^{(i)}, H_2^{(i)}, \ldots, H_N^{(i)})$. The content embedding of item attributes, denoted as $q^j_{i_c}$, is composed of the one-hot encoding of item attributes and the content embedding of the attributes, and can be formulated as follows:

$$q_{i_c}^{j} = B_{i_c} H_{i_c}^{(i)} \tag{6}$$

Where $1 \le c \le N$, B_{i_c} contains the embeddings of all the content in the item attributes as column vectors. Figure 3 illustrates the overall architecture of the recommender network on the client-side. The recommender network is primarily responsible for predicting user-item ratings, where the user-item embeddings q_{i_c} are formed by concatenating the transformed content embeddings of item attributes, and can be expressed as follows:

$$q_{i_c} = \sigma(w_i(\hat{q}_{i_1}^{j} \oplus \ldots \oplus \hat{q}_{i_N}^{j}) + b_i) \tag{7}$$

Where $\hat{q}_{i_N}^{j} = \sigma(w_N q_{i_N}^{j} + b_N)$, $\sigma(\cdot)$ represents the activation function ReLU. $\oplus$ denotes vector concatenation, w_i stands for the weights, and b_i represents the biases. The user embedding task is primarily designed to capture user preferences towards item attribute content. We employ residual block with attention to facilitate user embeddings to selectively capture user preferences towards item attribute content. The formulation is as follows:

$$\hat{q}_{i_c} = q_{i_c}^{a} + q_{i_c} \tag{8}$$

Where $q_{i_c}^{a} = Atention(p_{u_c}, a)$, $Atention()$ is an attention layer. The structure of residual block with attention in the model can be described as follows: the residual block with attention receives two inputs, namely user features and item attribute features. Initially, the attention weights are computed by measuring the similarity between user features and item attribute features. Subsequently, these attention weights are applied to the item attribute features, resulting in a weighted sum that yields attention-weighted item attribute features. The attention-weighted item attribute features are then residue-connected with the original item attribute features. Finally, the features obtained after the residual connection are normalized and activated. a contains the content embeddings of item attributes as column vectors. The user's rating of the item is predicted by the neural network $r(\cdot)$, and the formulation is as follows:

$$\hat{R}_{u_c,i_c} = r(\hat{q}_{i_c}, p_{u_c}; \beta) \tag{9}$$

Where β represents the parameters of the neural network $r(\cdot)$ including weights and biases. The loss function of the recommender network uses a personalized loss function, and the formulation is as follows:

$$L(R_{u_c,i_c}, \hat{R}_{u_c,i_c}) = | \, (R_{u_c,i_c} - \hat{R}_{u_c,i_c}) \, |^2 \tag{10}$$

Where R_{u_c,i_c} represents the original ratings in the dataset on client i, and $\hat{R}_{u_c,i_c}$ represents the predicted ratings by the recommender network. The HNG on the central server is responsible for generating all parameters of the recommender network on each client. The parameter set $\Phi = \{\theta_1, \theta_2, \ldots, \theta_N\}$ represents all the parameters in the recommender network on the client, where $\theta_i = \{w_i, b_i, \beta\}$.

4.3 Reinforcement Selector

The RLS is deployed on the server and undertakes the primary role of client selection for engagement in federated

communication within each round t. The central aim revolves around optimizing the mean test accuracy across clients, simultaneously minimizing cumulative processing latency and communication cost. Each client i decides whether to participate in the federated communication based on the decisions made by RLS. RLS consists of two components, namely Multi-agent and Data Center. The Multi-agent is a multi-layer perceptron. During the training process of Multi-agent, each agent in Multi-agent receives the current state and predicts actions. The first round of local training in the training process is referred to as test training. Subsequently, the central server calculates the total reward based on the average test accuracy of clients, communication costs, and total processing delay. Then, each agent in Multi-agent uses a deep neural network (DNN) for training to maximize the total reward.

The state of each agent i in Multi-agent can be composed of six components: processing delay of test training, historical values of communication delay, test loss, communication costs, and local training dataset size. The formulation is as follows:

$$s_i^t = [h_{t,i}^d, g_{t,i}^c, l_i^t, C_i^t, k_i] \tag{11}$$

Where $h_{t,i}^d$ represents the processing delay of test training, and $g_{t,i}^c$ represents the historical values of communication delay. To predict the current training delay and communication delay for client i, each agent i in the Multi-agent has access to the historical test training delay $h_{t,i}^d = [\mathrm{h}_{t-\Delta H_d,i}^d, \ldots, \mathrm{h}_{t,i}^d]$ and communication delay $g_{t,i}^c = [\mathrm{g}_{t-\Delta H_c,i}^c, \ldots, \mathrm{g}_{t,i}^c]$. Where $\mathrm{h}_{\mathrm{t,i}}^{\mathrm{d}}$ represents the processing delay of agent i in the t-th round of test training, and $\mathrm{g}_{t,i}^c$ represents the communication delay from the client model to the central server for the update direction in the t-th round. Additionally, ΔH_d and ΔH_c indicate the size of historical information for test training delay and communication delay, respective. Moreover, l_i^t represents the test training loss of agent i in the t-th round, which is uploaded to the central server's Multi-agent during the t-th round of Multi-agent training process. This uploaded loss l_i^t helps predict the deviation of agent i's recommended model update. Furthermore, C_i^t denotes the communication cost for client i in the t-th round, and k_i represents the size of the local training dataset on client i.

In each round of the training process, each agent in Multi-agent prevents half of the clients selected from the pool of edge devices from participating in the federated learning process if their test losses exceed the average test loss. Each agent i in Multi-agent makes a decision regarding the premature termination of client i's involvement in federated learning process, and the formulation for this decision is as follows:

$$d_i^t \in \{0,1\} \tag{12}$$

Where d_i^t represents whether client i participates in the federated learning process in round t. $d_i^t = 1$ indicates the client will continue to participate in the next round of training after the test training. Let $G_{t,i}^c$ represent the local training time for client i in round t. The total processing delay for round t can be represented as:

$$G_t = max_{1 \le i \le N}(G_{t,i}^c + g_{t,i}^c)d_i^t \tag{13}$$

Where $g_{t,i}^c$ represents the communication delay of uploading the update direction from the client model to the central server in round t. $C_t = \sum_i C_i^t d_i^t$ represents the total communication cost in round t. The reward function r_t needs to reflect the changes in processing delay, communication cost, and test accuracy after the client executes the RLS

decision, and it can be formulated as follows:

$$r_t = a_1[F(Test(t) - Test(t-1))] - a_2 G_t - a_3 C_t \tag{14}$$

Where $Test(t)$ represents the average test accuracy of the client on the global test dataset after the last round of training. $F(\cdot)$ is a utility function used to quantify the preference for different outcomes, ensuring that the reward continues. a_1, a_2, a_3 is a coefficient that controls the importance of the objective. $G_t = max_{1\le i\le N}(\mathrm{h}_{t,i}^d) + max_{1\le i\le N, d_i^t=1}(g_{t,i}^{local} + G_{t,i}^c)$ is the processing delay in round t, and $g_{t,i}^{local}$ is the time required for client i to complete local training.

Algorithm 1: Sever in each communication round

Input: Rounds: n
Local steps: s
HNG learning rate: α

Output: Parameters of the recommender network: θ_i

for i = 1, . . . , n **do**
  while $l_i^t \le \frac{1}{N}\sum_{1\le i\le N} l_i^t$ **do** //client loss is lower than average test loss
    Random selection of client i
    set $\theta_i = f_h(p_{\hat{u}_c}, \phi_h)$
    for j = 1, . . . , s **do**
      Random minimization batch $S \subset D_i$
      ClientUpdate
    end for
    $\phi_h = \phi_h - \alpha\nabla_{\phi_h}\theta_i^T \Delta\theta_i$ //update parameters
end for

The selected edge devices then receive the generated model parameters from the HNG. Subsequently, the client models perform one round of test training and send the test losses to the Multi-Agent, as well as processing delay and communication delay information to the Data Center. After receiving all the test losses from the clients, the RLS combines the test losses with processing delay, communication delay, and other information to generate client decisions for determining whether clients will participate in the next round of federated learning. Next, the chosen client devices perform model updates and transmit the update directions back to the HNG. The HNG aggregates all client update directions to achieve model sharing among clients.

The pseudo-code for FedRL is presented in Algorithm 1 and Algorithm 2. In Algorithm 1, the input consists of the number of iterations n, the local steps s, and the learning rate α of the HNG. The output is the parameters θ_i of the recommender network. Algorithm 1 is a server-side loop that generates the parameters of the client's recommender model and collects their update directions for further updates. In Algorithm 2, the input consists of the client's learning

rate β, the client's data $u_j^{(i)}, i_j^{(i)}, R_j^{(i)}$, and the learning rate λ, μ of the recommended network. The output is the update direction $\Delta\theta_i$ of HNG and the model's predicted user ratings $\hat{R}_{u_c,i_c}$. Algorithm 2 is a client-side loop that updates the partial embedding of the client's model and predicts item ratings.

Algorithm 2: ClientUpdate

Input: Client learning rate: β
Data for client i: $u_j^{(i)}, i_j^{(i)}, R_j^{(i)}$
Learning rate of recommender network: λ, μ

Output: Direction for updating HNG: $\Delta\theta_i$
User ratings of model prediction: $\hat{R}_{u_c,i_c}$

for $u_j^{(i)} \in U_t$ **do**
  Initialize parameters in θ_i
  Set $\tilde{\theta}_i = \theta_i$
  for $(\mathbf{u_j^{(i)}}, \mathbf{i_j^{(i)}}, \mathbf{R_j^{(i)}}) \in \mathbf{K}$ **do**
    $\forall \chi \in \theta_i : \chi := \chi - \lambda \nabla_\chi L(R_{u_c,i_c}, \hat{R}_{u_c,i_c})$ //update client model parameters
    $p_{u_c^*} := p_{u_c^*} - \alpha \nabla_{p_{u_c^*}} L(R_{u_c,i_c}, \hat{R}_{u_c,i_c})$ //update the basis for user preference weights
  end for
  for $(u_j^{(i)}, i_j^{(i)}, R_j^{(i)}) \in N$ **do**
    Predict $\hat{R}_{u_c,i_c}$ by the recommender network
    $B_{i_c} = B_{i_c} - \mu \nabla_{B_{i_c}} L(R_{u_c,i_c}, \hat{R}_{u_c,i_c})$
    $p_{u_c} = p_{u_c} - \mu \nabla_{p_{u_c}} L(R_{u_c,i_c}, \hat{R}_{u_c,i_c})$
    $\tilde{\theta}_i = \tilde{\theta}_i - \beta \nabla_{\tilde{\theta}_i} L_i(S)$ //client model performs local optimization
  end for
  $\Delta\theta_i = \tilde{\theta}_i - \theta_i$ //direction for updating the client model
end for

5 Experiments

Next, we present our research questions (Section 5.1), our experimental setup (Section 5.2), our evaluation metrics (Section 5.3), and finally, detailed evaluation results (Section 5.4).

5.1 Research Questions

In this research, we aim to tackle the subsequent research questions:

- RQ1: How does the recommender performance of FedRL compare to state-of-the-art federated recommender systems?
- RQ2: How does the recommender performance of FedRL compare to fully decentralized counterparts?

－RQ3: What is the impact of hyperparameters on FedRL?

－RQ4: What are the contributions of the key components in FedRL?

－RQ5: Can FedRL achieve efficient communication?

－RQ6: How does FedRL perform in terms of adapting to new users?

Table 2. Statistics on datasets.

Datasets	users	items	ratings	sparsity
Movielens1M	6,040	3,706	1,000,209	95.53%
Ciao	7,375	105,096	282,619	99.95%
Douban-Movie	2,964	39,695	894,888	99.23%
Amazon Movies	15,067	69,629	877,736	99.92%

5.2 Experiment Setup

5.2.1 Datasets

We conduct experiments using three widely used datasets, including:

- Movielens1M [53]. A commonly used dataset for movie ratings, it includes movie rating data from Movielens. Besides user ratings for movies, it also includes additional details about movies and users, such as movie IDs, titles, genres, etc.
- Ciao [54]. It originates from the Ciao social networking website and includes user reviews of products, product attribute information, user social relationships, and some other metadata.
- Douban-Movie [55]. It originates from Douban, a well-known Chinese movie social networking website, and includes movie information, user ratings, reviews, and social relationships.
- Amazon Movies [77]. It is a widely used movie rating dataset provided by Amazon. The dataset comprises a substantial amount of user ratings and review information for movies.

The detailed data for each dataset is presented in Table 2. Conducting experiments on these three datasets with different sizes and data sparsity enables a comprehensive assessment of the model's performance.

5.2.2 Baselines

We adopt two benchmarks for comprehensive comparison: centralized recommender systems and federated recommender systems. Centralized recommender systems necessitate users to transmit their private data to a central server for processing, which has the potential to compromise user privacy. Most existing centralized recommender methods need to handle massive user behavior and item data, leading to increased computational and storage burdens.

Additionally, user behavior data often needs to be retrieved from distributed storage, involving a significant amount of data transmission and leading to increased network overhead.

Centralized Recommender System

- DDL [56]:A hash-based discrete collaborative filtering framework is proposed, which maps users and items into a Hamming space to calculate user preferences for items based on Hamming distance. Leveraging deep learning techniques, a model known as the Deep Belief Network (DBN) is employed to derive informative item representations from the content-related information associated with the items.
- NCF [57]:A neural network-based recommender system model is proposed, which combines collaborative filtering and deep learning approaches. It focuses on implicit feedback and has the capability to predict user preferences for unknown items based on users' historical behaviors and item attribute features. Employing a multi-layer perceptron, the model acquires the ability to grasp the interaction function existing between users and items.
- SASRec [58]:A self-attention-based sequential recommender model is proposed. By employing the attention mechanism, it gains the ability to comprehend the significance of relationships among distinct items within the sequence. The sequence depicting user behavior is transformed into compact vector representations. In this process, the self-attention mechanism comes into play, facilitating the computation and modeling of item weights and interactions within the sequence. This, in turn, permits the encapsulation of long-term semantics.
- URL [78]:A reinforcement learning-based recommender system model is proposed. The training of the reinforcement learning agent is enhanced through auxiliary tasks. User response modeling is employed to augment the learning of state and action representations for the recommender agent.
- RecVAE [59]:A neural network-based autoencoder collaborative filtering model is proposed. It incorporates a variational autoencoder with a latent code that follows a prior distribution. The training is accomplished through alternating updates of the encoder and decoder. The model allows for training the encoder with corrupted data.
- EX^3 [79]:A recommender system based on item attribute information is proposed. By analyzing users' historical behaviors, critical attributes are learned, leading to the derivation of a recommended set of items. Utilizing multi-stage learning, the system adaptively selects recommended items for users and determines crucial attributes.
- DAVE [60]:A robust recommender model is proposed, which utilizes an autoencoder to capture users' true preferences from noisy data. It provides personalized denoising for different users and items, capturing multi-modalities of the embedding space. The model enhances robustness by modeling the variance of the inference distribution to handle noise.

Federated Recommender System

- PrivRec [61]:A DNN-based federated recommender model is proposed, which introduces a first-order meta-learning approach to achieve personalization on individual devices. It uses a two-step training method to compensate for the loss incurred during model updates with added noise. A user-level differential privacy model is developed to protect users from being identified by attackers.

- MetaMF [62]:A federated matrix factorization framework is proposed, which leverages meta-networks to generate private item embeddings and recommender models. It utilizes a collaborative memory model to gather useful information. A dimensionality enhancement strategy is designed to selectively generate item embeddings with targeting.
- LightFR [63]:A matrix factorization-based federated recommender system framework is proposed. Harnessing the power of hashing methods, it employs techniques to formulate high-quality binary codes within the context of federated setups. Furthermore, a federated algorithm for discrete optimization is utilized to cooperatively train model parameters across both the server and clients.
- FedRecon [64]:A personalized federated learning approach is proposed and evaluated using matrix factorization. It employs partial local federated learning, where the model is divided into global and local parameters, and the local parameters never leave the client devices. This enables specific parameters to be used for training sensitive users.
- PFedRec [65]:A personalized federated recommender framework is proposed, which puts forth a dual personalization mechanism, designed to acquire nuanced personalized insights for both users and items. This framework enables the refinement of item embeddings, individualizing them for each user. The outcome is the creation of user-specific perspectives on item representations. The framework is designed to learn lightweight models deployed on the client-side.

5.2.3 Implementation details

In the federated recommender system, each user is considered as a local client. For each user, the data is divided into three parts: 80% for training, 10% for testing, and 10% for validation. Hyperparameters are adjusted using grid search. The user batch size in each training round is searched from {16, 32, 64, 128, 256}, the learning rate is searched from {0.001, 0.01, 0.1} based on the validation set performance, and the embedding size is adjusted from {4, 8, 16, 32, 64}. The coefficient for the target importance in the reward function of RLS is set to $[a_1, a_2, a_3] = [1,0.1,0.1]$. Other model details for the baseline methods follow the original papers. During the training process, an early stopping technique is implemented. If the performance of the recommender model on the validation set does not display any enhancement over five consecutive instances, the training procedure is halted. All experimental trials are iterated ten times, and subsequently, the averaged outcomes along with their corresponding standard deviations are presented.

5.3 Evaluation Metrics

For the assessment of result accuracy and diversity, we employ evaluation metrics such as the HR (Hit Rate) and $NDCG$ (Normalized Discounted Cumulative Gain), which are commonly used in recommender systems [66, 67, 68, 69]. These metrics serve as indicators to gauge the performance of the recommender system. HR is used to measure whether the recommended list contains items that the user is truly interest in. The calculation of HR involves counting the occurrences of user's interested items in the recommender list. if an item finds its place within the recommender list, a corresponding assignment of 1 is allocated; contrarily, a value of 0 is designated. Consequently, the ultimate metric,

referred to as HR, stands for the fraction of instances marked as 1s. Meanwhile, the parameter denoted as HR, encompasses a range spanning from 0 to 1. A higher value of HR corresponds to an enhanced performance of the recommender system. In this paper, the formula for HR is as follows:

$$HR@10 = \frac{|\ GT_{10}^{hit}\ |}{|\ GT_{10}\ |} \tag{15}$$

Where $|\ GT_{10}\ |$ represents the length of the recommender list, which is ten in this case, and $|\ GT_{10}^{hit}\ |$ represents the number of correct hits among the ten predictions. $NDCG$ serves as a gauge of ranking excellence, encompassing considerations of both the sequence and ranking specifics inherent within the outcomes yielded by the recommender. It is the normalization of DCG score with respect to $IDCG$ score. DCG considers both the relevance scores of the recommended items and their ranking positions, and it penalizes items that appear at the bottom of the list. On the other hand, $IDCG$ represents the discounted cumulative gain of the maximum possible ranking of the recommender results. The relevant formulas for DCG and $IDCG$ are as follows:

$$DCG@10 = \sum_{i=1}^{GT_{10}} \frac{rel(i)}{log_2(i+1)} \tag{16}$$

$$IDCG@10 = \sum_{i=1}^{GT_{10}^{*}} \frac{rel(i)}{log_2(i+1)} \tag{17}$$

$$NDCG@10 = \frac{DCG@10}{IDCG@10} \tag{18}$$

Where $rel(i)$ represents the relevance score of item i, and GT_{10}^{*} denotes the ideal list of items. We calculate HR and $NDCG$ for each user and report their average values and standard deviations.

5.4 Performance Analysis

In this section, we present comprehensive experimental results comparing FedRL with centralized recommender system methods and state-of-the-art federated recommender systems. Specifically, we compare FedRL with twelve other recommender system methods in terms of overall recommender performance. Subsequently, we conduct sensitivity analysis, validate the communication efficiency of FedRL, perform ablation experiments to analyze the specific contributions of our proposed method, and finally, engage in a cold start experiment.

5.4.1 Overall Performance (RQ1 and RQ2)

We compare FedRL with various models, including seven centralized recommender system methods and five federated recommender system methods. Within Table 3 and Table 4, an encapsulation of the experimental outcomes for $HR@10$ and $NDCG@10$ across four extensively utilized datasets is presented. The initial aspect delves into the assessment of the relative efficacy between methodologies applied in both centralized recommender systems and FedRL.

From these results, we draw the following observations:

- In the centralized recommender system methods, the superiority in performance exhibit by the NCF model in comparison to the DDL model underscores the benefits derive from the utilization of multi-layer neural networks for capturing implicit attributes associated with users and items. In contrast, the DDL model employs discrete deep learning methods, which often use binarization techniques to handle discretized input data. Moreover, the better performance of SASRec compared to NCF can be attributed to its modeling of the entire user sequence and the introduction of a hierarchical self-attention mechanism, allowing the model to learn more complex item transitions. URL outperforms SASRec because it introduces auxiliary tasks to assist the agent in more targeted exploration.
- Methods based on variational autoencoders, such as RecVAE, achieve better results than URL because they introduce a compound prior distribution for the variational encoder to stabilize training. EX^3 achieves superior performance compared to RecVAE by employing multi-stage learning to assist the model in better handling diverse user scenarios and requirements. In most cases, DAVE outperforms EX^3 because it combines variational inference with adversarial training.
- Furthermore, compare to the aforementioned centralized recommender systems, FedRL achieve the best performance. The improvements in metric $HR@10$ on the Movielens1M, Ciao, Douban-Movie and Amazon Movies datasets were 5.1%, 3.5%, 2.6% and 6.7%, respectively. Additionally, the improvements in metric $NDCG@10$ were 1.6%, 2.7%, 6.8% and 3.7%, respectively. We attribute this success to the HNG, which facilitates the sharing of client network parameters, enabling collaborative learning among participants.

Table 3. Results of FedRL compared to a centralised recommender system approach on four datasets.

Methods	Movielens1M		Ciao	
	$HR@10$	$NDCG@10$	$HR@10$	$NDCG@10$
DDL	0.5027±0.0013	0.2751±0.0008	0.4315±0.0025	0.2302±0.0028
NCF	0.5286±0.0021	0.2893±0.0010	0.4561±0.0034	0.2472±0.0025
SASRec	0.5849±0.0012	0.3562±0.0015	0.5189±0.0026	0.2943±0.0022
URL	0.6148±0.0013	0.3791±0.0011	0.5507±0.0023	0.3241±0.0021
RecVAE	0.6873±0.0010	0.3984±0.0021	0.6183±0.0021	0.3518±0.0031
EX^3	0.6915±0.0012	0.4082±0.0010	0.5864±0.0024	0.3285±0.0025
DAVE	0.6997±0.0011	0.4108±0.0008	0.6059±0.0025	0.3425±0.0034

FedRL	0.7354±0.0015	0.4272±0.0009	0.6401±0.0014	0.3613±0.0018
Improve	5.1%	1.6%	3.5%	2.7%
Methods	Douban-Movie		Amazon Movies	
	$HR@10$	$NDCG@10$	$HR@10$	$NDCG@10$
DDL	0.3129±0.0015	0.1985±0.0017	0.5431±0.0013	0.3140±0.0015
NCF	0.3505±0.0014	0.2143±0.0021	0.5737±0.0018	0.3205±0.0020
SASRec	0.3960±0.0025	0.2511±0.0032	0.6221±0.0014	0.3854±0.0016
URL	0.4321±0.0019	0.2630±0.0020	0.6573±0.0012	0.4124±0.0013
RecVAE	0.4726±0.0027	0.2784±0.0024	0.6951±0.0013	0.4175±0.0011
EX[3]	0.4772±0.0025	0.2861±0.0022	0.7141±0.0011	0.4236±0.0013
DAVE	0.4839±0.0024	0.2951±0.0018	0.7326±0.0012	0.4583±0.0015
FedRL	0.4967±0.0012	0.3153±0.0015	0.7816±0.0013	0.4754±0.0010
Improve	2.6%	6.8%	6.7%	3.7%

- Simultaneously, it aids in enhancing the model's generalization ability, enabling better adaptation to new users and items, thereby improving overall recommender performance. Furthermore, the introduction of item attribute content information expands the model's feature space, enabling a more comprehensive description of item features and content. This assists the model in capturing the relationships between users and items more comprehensively.
- Subsequent, our attention turns to the experimental scrutiny of the federated recommender system approaches in contrast to our novel FedRL method. Table 4 displays the experimental outcomes, revealing a number of notable observations.
- The recommender performance of PrivRec and MetaMF often falls short of centralized recommender system models, while LightFR, FedRecon, and PFedRec models perform comparably to centralized recommender system models. Our observations stem from the inherent design of the federated learning structure. In this architecture, the imperative focus on safeguarding privacy and local data confinement inhibits the federated framework's direct

access to localized data. As a consequence, the framework's capacity to encapsulate the global structure is constrained.

Table 4. Results of FedRL compared to the Federal Recommender System approach on four datasets.

Methods	Movielens1M		Ciao	
	HR@10	$NDCG$@10	HR@10	$NDCG$@10
PrivRec	0.4729±0.0018	0.2541±0.0012	0.4207±0.0034	0.2315±0.0043
MetaMF	0.4853±0.0023	0.2697±0.0015	0.4192±0.0032	0.2218±0.0036
LightFR	0.5356±0.0017	0.3015±0.0008	0.4631±0.0025	0.2539±0.0027
FedRecon	0.6218±0.0014	0.3502±0.0021	0.5326±0.0021	0.3240±0.0024
PFedRec	0.7064±0.0019	0.4153±0.0017	0.5942±0.0031	0.3501±0.0025
FedRL	0.7354±0.0015	0.4272±0.0009	0.6401±0.0014	0.3613±0.0018
Improve	4.1%	2.7%	7.7%	3.2%
Methods	Douban-Movie		Amazon Movies	
	HR@10	$NDCG$@10	HR@10	$NDCG$@10
PrivRec	0.3162±0.0024	0.1891±0.0022	0.5065±0.0015	0.2843±0.0011
MetaMF	0.3293±0.0027	0.1938±0.0034	0.5281±0.0019	0.3045±0.0012
LightFR	0.3137±0.0021	0.1921±0.0029	0.5524±0.0014	0.3357±0.0009
FedRecon	0.4594±0.0018	0.2703±0.0025	0.6514±0.0011	0.3953±0.0018
PFedRec	0.4785±0.0023	0.3056±0.0019	0.7423±0.0016	0.4569±0.0013
FedRL	0.4967±0.0012	0.3153±0.0015	0.7816±0.0013	0.4754±0.0010
Improve	3.8%	3.2%	5.3%	4.0%

- LightFR's recommender performance is significantly better than PrivRec and MetaMF. We find that the reason for this is that PrivRec's first-order meta-learning method is prone to getting stuck in local optima when dealing with complex non-convex problems. Conversely, the utilization of the meta-recommender module by MetaMF for generating private item embeddings and suggesting models renders it susceptible to the choice of the search space. In contrast, LightFR leverages hash technology to generate binary codes, which effectively handles non-convex problems.
- FedRecon achieve better results than LightFR, and we find that this is because it retrains user embeddings from scratch at the beginning of each pair of rounds instead of continuing to inherit the user embeddings from the previous round. Among the five federated recommender system baselines, PFedRec outperforms the other four baselines in most cases. Our observation arises from the utilization of a dual personalization mechanism within PFedRec. This mechanism facilitates the acquisition of intricate user and item personalization, thereby enhancing the capacity for precise relationship alignment between users and items. Consequently, this mechanism fosters the delivery of personalized recommender outcomes that exhibit a heightened alignment with individual user preferences.

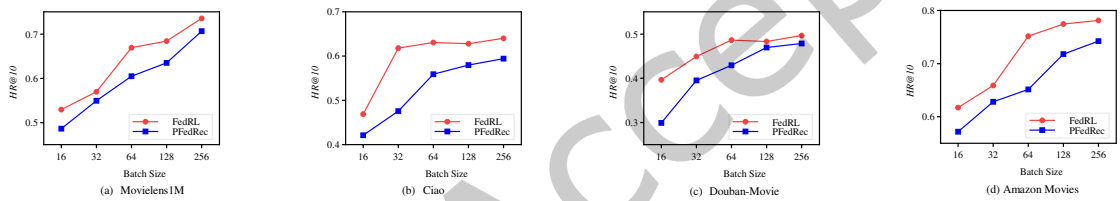

Figure 4: Performance Analysis of $HR@10$ with Varying Batch Sizes across Four Datasets.

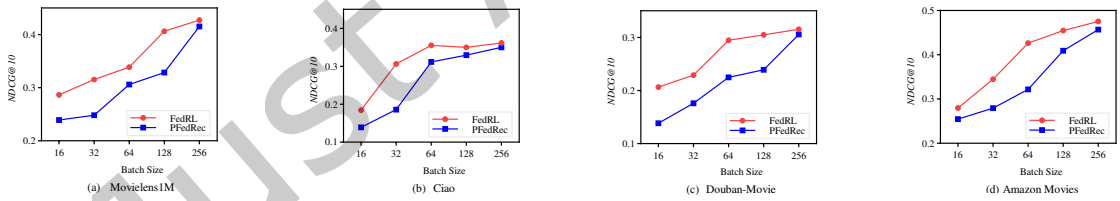

Figure 5. Performance Analysis of $NDCG@10$ with Varying Batch Sizes across Four Datasets.

- Our proposed FedRL consistently achieve the best recommender performance in all scenarios. Compared to the centralized recommender system baseline, FedRL achieve improvements of 2.7%, 3.2%, 3.2%, and 4.0% in terms of metric $NDCG@10$ on Movielens1M, Ciao, Douban-Movie and Amazon Movies datasets, respectively. Compared to the federated recommender model baseline, FedRL achieved improvements of 4.1%, 7.7%, 3.8%, and 5.3% in terms of metric $HR@10$ on the same datasets, respectively. These results further validate that our method can maintain excellent recommender performance in scenarios with incomplete initial information and

data sparsity. We attribute this success to HNG, which assists the client models in dynamically adjusting their model parameters, thus maintaining the model's timeliness and accuracy and improving the recommender effectiveness. Simultaneously, RLS assists the model in selecting edge devices more effectively in each round of the federated learning process, enhancing the overall model performance.

5.4.2 Sensitivity Study (RQ3)

In this section, we focus on analyzing the impact of FedRL hyperparameters, including user batch size, embedding size, and the learning rate of the hypernetwork.

5.4.2.1 User Batch Size

In this section, we analyze the impact of user batch size on the recommendaer performance of FedRL. We explore a range of user batch sizes, including {16, 32, 64, 128, 256}. Intuitively, selecting a smaller user batch size may increase the number of communication rounds required for training the model on the server, while choosing a larger batch size may lead to increased memory consumption during model training, potentially causing memory issues. In Figure 4 and Figure 5, we present the values of $HR@10$ and $NDCG@10$ for FedRL and PFedRec on the Movielens1M, Ciao, Douban-Movie, and Amazon Movies datasets, and we draw the following observations:

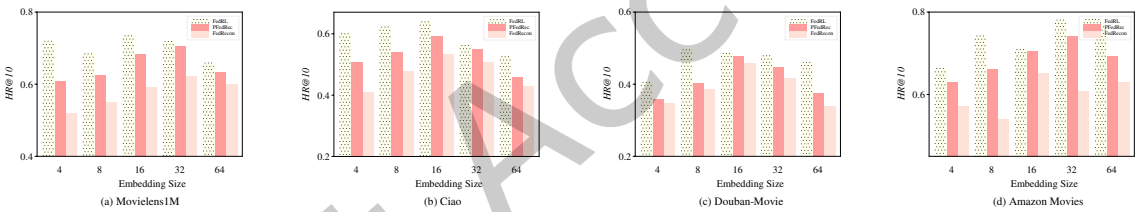

Figure 6. Performance Evaluation of $HR@10$ with Various Embedding Sizes on Four Datasets.

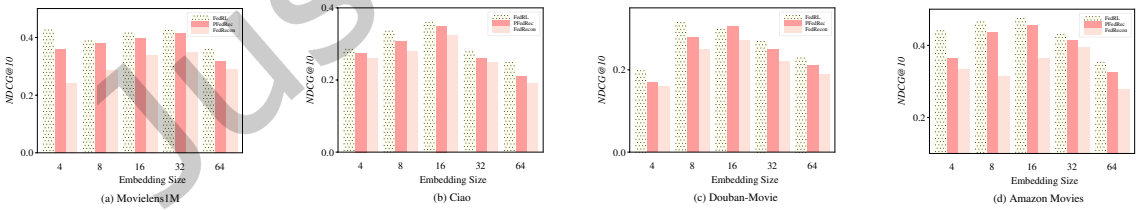

Figure 7. Performance Evaluation of $NDCG@10$ with Various Embedding Sizes on Four Datasets.

- The performance of FedRL surpasses PFedRec. On all four datasets, the values of $HR@10$ and $NDCG@10$ in FedRL consistently exceed those in PFedRec. This can be attributed to the powerful dynamic generation capability of the super-network generator. The performance of FedRL improves with an increase in the user batch size on the

dataset. We observe that larger batch sizes can provide more interaction data, which helps the model better learn user preferences and the relationships between items. As the user batch size increases, more accurate parameter updates can be obtained, thereby improving the model's generalization ability.

- In the Ciao dataset, we observe a more significant improvement in model performance when the user batch size increased from 16 to 32. This can be attributed to the dataset's sparsity, where the model can only see a small number of data samples during each parameter update. This limited visibility may not be sufficient for the model to effectively learn the underlying features and patterns in the data. As the batch size augments, the model gains the capability to access more precise global information, consequently resulting in an enhancement in performance quality.

5.4.2.2 Embedding Size

This section investigates the impact of embedding sizes on the performance of FedRL. Figure 6 and Figure 7 present the values of $HR@10$ and $NDCG@10$, respectively, for FedRL on three datasets, with embedding sizes ranging from {4, 8, 16, 32, 64}. For the sake of comparative analysis, we incorporate two noteworthy baselines for reference: FedRecon and PFedRec. From these results, we draw the following observations:

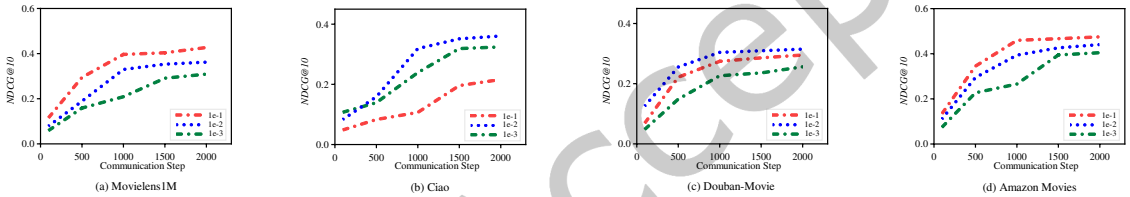

Figure 8. Performance effects of $NDCG@10$ on four datasets for different learning rate.

- The FedRL approach exhibits superior performance compared to previous federated methods across various embedding sizes. We observe that FedRL exhibits superior adaptability to various recommender scenarios, attributed to its dynamic generation of user network parameters by the HNG, allowing the model to flexibly accommodate the features and interests of different users, achieving improved personalized adaptation. Simultaneously, it facilitates the model in promptly adapting to changes in user behavior and data distribution, enhancing the model's performance under different data distributions. PFedRec demonstrates better performance than FedRecon, providing evidence for the effectiveness of learning fine-grained features of users and items.
- Compare to FedRecon and PFedRec, FedRL exhibits smaller fluctuations as the embedding size varies. The results indicate that FedRL demonstrates better robustness with varying embedding sizes and possesses superior feature representation capabilities, enabling it to capture useful information from the data across different embedding sizes. This highlights the effectiveness of utilizing the HNG to dynamically generate model parameters. The HNG achieves flexibility and diversity across different edge devices.

5.4.2.3 Learning Rate

The learning rate determines the step size for updating model parameters during the training process, affecting the convergence speed and performance of the model. It also impacts the communication rounds in the federated recommender system. Fewer communication steps can reduce communication overhead and mitigate instability among devices. The study focus on examining the influence of hypernetwork learning rates on the performance of FedRL, using the evaluation metric $NDCG@10$. The hypernetwork learning rates are chosen from the range $\{0.001, 0.01, 0.1\}$, and the results are reported in Figure 8. Our findings are as follows:

- As the number of communication rounds increases, FedRL gradually levels off across the four training datasets. These results indicate that the federated learning framework utilize by FedRL effectively transfers information gradients, leading to convergence. In Figure 8, it can be observed that, under appropriate learning rates, the performance curve of FedRL remains relatively stable, suggesting that our model exhibits good robustness and adapts well to noise and variations in the dataset.

  The performance curves of the model gradually flatten under three different learning rates, indicating the strong generalization ability of FedRL, capable of exhibiting good performance in diverse environments. Additionally, after a certain number of communication steps, the performance curves of FedRL tend to stabilize, demonstrating the model's robustness and reliability in the face of client heterogeneity and varied training data distributions.
- Different datasets exhibit varying preferences for the learning rate. An illustrative example pertains to the Movielens1M dataset, where in the optimal learning rate is established at 0.1. In contrast, on the Ciao dataset, the most favorable learning rate is determined to be 0.01. It is noteworthy that the convergence speed across distinct datasets is notably influenced by the selected learning rate. Excessively high learning rates can lead to an unstable optimization process. To illustrate, considering the Ciao dataset, configuring the learning rate as 0.1 triggers a situation where the model alternates cyclically within the parameter space. This phenomenon subsequently leads to suboptimal convergence performance.

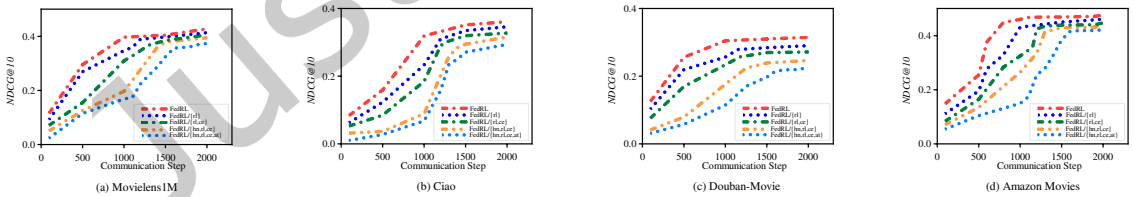

Figure 9. The performance of different variants of FedRL is evaluated in terms of $NDCG@10$ across the four datasets at various communication steps.

Table 5. The $HR@10$ and $NDCG@10$ performance of FedRL variants on the four datasets.

Variants	Movielens1M		Ciao	
	$HR@10$	$NDCG@10$	$HR@10$	$NDCG@10$
FedRL/{hn,rl,ce,at}	0.6712	0.3735	0.5648	0.2941
	(-8.7%)	(-12.6%)	(-11.8%)	(-18.6%)
FedRL/{hn,rl,ce}	0.6883	0.3948	0.5782	0.3159
	(-6.4%)	(-8.2%)	(-9.7%)	(-12.9%)
FedRL/{rl,ce}	0.7026	0.4053	0.5924	0.3274
	(-4.5%)	(-5.1%)	(-7.4%)	(-9.4%)
FedRL/{rl}	0.7141	0.4135	0.6186	0.3462
	(-2.8%)	(-3.2%)	(-3.4%)	(-4.2%)
FedRL	0.7354	0.4272	0.6401	0.3613
Variants	Douban-Movie		Amazon Movies	
	$HR@10$	$NDCG@10$	$HR@10$	$NDCG@10$
FedRL/{hn,rl,ce,at}	0.3727	0.2281	0.7249	0.4216
	(-25.0%)	(-27.7%)	(-7.3%)	(-11.3%)
FedRL/{hn,rl,ce}	0.3956	0.2468	0.7368	0.4374
	(-20.3%)	(-21.7%)	(-5.7%)	(-7.9%)
FedRL/{rl,ce}	0.4482	0.2715	0.7473	0.4471
	(-9.8%)	(-13.9%)	(-4.3%)	(-5.9%)
FedRL/{rl}	0.4753	0.2906	0.7652	0.4613
	(-4.3%)	(-7,8%)	(-2.1%)	(-3.0%)
FedRL	0.4967	0.3153	0.7816	0.4754

5.4.3 Ablation Study (RQ4)

The experiments in this section aim to validate the effectiveness of the components in FedRL. For simplicity, we use symbols to represent different variants of FedRL: "hn" denotes the method employing the HNG, "rl" refers to the approach utilizing the RLS, and "ce" indicates the utilization of content embeddings with item attributes as global Information, "at" indicates the use of residual block with attention. These symbols are used to specify the following variants of FedRL.

- The effect of using the HNG: We establish two model variants, namely FedRL/{hn,rl,ce} and FedRL/{rl,ce}. The FedRL/{hn,rl,ce} variant excludes hn, rl, and ce components from the FedRL architecture, while the FedRL/{rl,ce} variant builds upon FedRL/{hn,rl,ce} by introducing hn, which utilizes the HNG to generate

parameters for edge devices' networks. The performance of these two variants is presented in Table 5. Upon observation, it becomes evident that the variant employing the HNG (FedRL/{rl,ce}) exhibits significant performance improvements across all four datasets compared to FedRL/{hn,rl,ce}. This result underscores the effectiveness of using the HNG to adapt to varying data distributions and features on different devices, thereby enhancing the performance of federated recommender system.

- The effect of using residual block with attention: We established a model variant, FedRL/{hn,rl,ce,at}. This model compare with FedRL/{hn,rl,ce}, which does not include residual block with attention. Upon observation, it find that FedRL/{hn,rl,ce} outperforms FedRL/{hn,rl,ce,at} in all scenarios. This suggests that residual block with attention assists in embedding user preferences, focusing on carrying users' preferences for item attribute content. This contributes to enhancing the precision in characterizing user preferences.
- The impact of using the content embedding of item arrtibutes as global information: We establish a model variant called FedRL/{rl}, which considers the content embeddings of item attributes, such as item categories and tags, in contrast to FedRL/{rl,ce} that does not incorporate these attributes. The experimental results for these two variants are recorded in Table 5. Upon observation, it becomes evident that FedRL/{rl} outperforms FedRL/{rl,ce} in all cases. This finding highlights the rationale behind considering content embeddings that incorporate item attributes. The addition of content embeddings with item attributes enhances the model's ability to represent items more effectively, thereby better capturing the underlying features of the items.

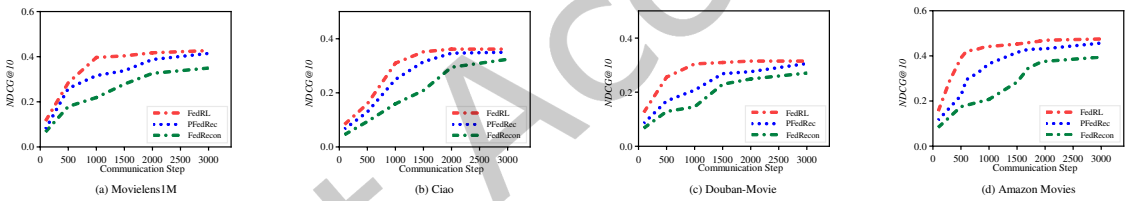

Figure 10. Performance effects of $NDCG@10$ on four datasets for different communication steps.

- The effect of using RLS: We establish a variant called FedRL/{rl}, which differs from FedRL by not utilizing the RLS. The experimental results record in Table 5 demonstrate that FedRL outperforms FedRL/{rl} across all four datasets. This finding emphasizes the importance of the RLS in the process of selecting client participants for federated learning. Engaging high-caliber clients in the training process holds the potential to alleviate the repercussions of noise and disparities in data on the model's overall performance. Furthermore, our analysis reveal that the use of the RLS leads to more significant performance improvements on the Ciao and Douban-Movie datasets. This observation might be attributed to the fact that these two datasets encompass a substantial number of users and items. The RLS can dynamically adjust client selection strategies based on feedback information from different environments, thereby yielding better performance in complex scenarios.

To further investigate the effectiveness of different components of FedRL in reducing communication overhead,

we report the values of $NDCG@10$ for FedRL variants as the communication steps vary on four datasets in Figure 9. We make the following observations:

- Compared to FedRL/{rl}, FedRL achieves convergence with fewer communication steps on all four datasets. The reduced communication steps help minimize the number of communications between the third-party server and the participants, indicating the effectiveness of the RLS in reducing communication overhead. The RLS determines which participants contribute effectively to the system's recommender performance and selects only those participants for the communication process of model updates. This reduces unnecessary communication overhead and improves communication efficiency.
- FedRL/{rl, ce} further eliminates the content embedding of item attributes based on FedRL/{rl}. From Figure 9, it can be observed that FedRL/{rl, ce} and FedRL/{rl} have similar communication steps at which the curves start to converge on the four datasets. However, there is a significant difference in the values of $NDCG@10$. Therefore, whether to use content embedding of item attributes as global information will impact the system's recommender performance but provides little help in reducing communication overhead.

  Compared to FedRL/{rl, ce}, FedRL/{hn, rl, ce} requires more communication steps for the curves to converge on all four datasets. This indicates that the HNG is helpful in reducing communication overhead. The HNG dynamically generates parameters for client models during the communication process of federated learning, effectively reducing the parameter size of client models and thus reducing communication overhead. Comparison between FedRL/{hn,rl,ce,at} and FedRL/{hn,rl,ce} indicates similar convergence steps in terms of communication steps across four datasets, but notable differences in $NDCG@10$ values. This suggests that the use of "at" significantly influences model performance, with minimal impact on communication overhead.

Table 6. Cold start performance comparison of FedRL across four datasets.

η	Movielens1M		Ciao	
	$HR@10$	$NDCG@10$	$HR@10$	$NDCG@10$
10%	0.7293±0.0025	0.4203±0.0020	0.6332±0.0028	0.3538±0.0029
20%	0.7320±0.0021	0.4233±0.0014	0.6363±0.0023	0.3571±0.0027
50%	0.7341±0.0018	0.4257±0.0011	0.6386±0.0019	0.3596±0.0024
100%	0.7354±0.0015	0.4272±0.0009	0.6401±0.0014	0.3613±0.0018
η	Douban-Movie		Amazon Movies	
	$HR@10$	$NDCG@10$	$HR@10$	$NDCG@10$
10%	0.4889±0.0029	0.3067±0.0031	0.7760±0.0019	0.4692±0.0022
20%	0.4923±0.0025	0.3105±0.0027	0.7785±0.0018	0.4719±0.0019
50%	0.4948±0.0020	0.3132±0.0022	0.7804±0.0015	0.4741±0.0015
100%	0.4967±0.0012	0.3153±0.0015	0.7816±0.0013	0.4754±0.0010

5.4.4 Communication Performance Study (RQ5)

This section aims to investigate the communication efficiency of FedRL. Figure 10 presents the variation of $NDCG@10$ values for FedRL across four datasets with different communication steps. For the purpose of comparison, we opte to consider two illustrative baselines, specifically FedRecon and PFedRec. Our observations are as follows:

- With an escalation in the count of communication rounds, the $NDCG@10$ performance curves of FedRL, FedRecon, and PFedRec exhibit a tendency to plateau. In Figure 10, it is evident that the $NDCG@10$ performance curve of FedRL reaches a plateau at a smaller number of communication rounds compared to FedRecon and PFedRec. Specifically, in Figure 10(a), FedRL achieves convergence at around 1000 communication steps, while PFedRec and FedRecon require approximately 2000 communication steps to achieve a similar balance. This finding indicates that FedRL exhibits superior communication efficiency. We attribute this improved efficiency to the dynamic client selection mechanism of the RLS, which intelligently chooses clients to participate in the federated process based on communication efficiency and resource constraints. This effectively reduces the model's communication overhead. Additionally, the HNG generates model parameters specific to individual clients, significantly reducing communication costs compared to traditional federated recommender models that require sending global parameters to all clients. Moreover, we employ the chain rule to further reduce the parameters transmitted from clients to the central server.
- In different datasets, the number of communication rounds required for the model to converge varies. Taking FedRL as an example, on the Movielens1M dataset, FedRL converges in approximately 1000 communication steps, whereas on the Ciao dataset, it requires around 1500 communication rounds to converge. We attribute this discrepancy to the characteristics and complexity of the datasets. The Ciao dataset is sparser compared to the Movielens1M dataset, and its data distribution is also sparser. Consequently, in scenarios involving the Ciao dataset, the model could necessitate a greater number of communication rounds to attain convergence. The sparsity and distribution of data significantly influence the convergence behavior of the model.

5.4.5 Cold Start Study (RQ6)

To conduct cold start experiments in scenarios with initial incomplete information and sparse data, we randomly select a certain proportion $\eta \in \{10\%, 20\%, 50\%, 100\%\}$ of data from the training set to construct new training sets. Table 6 reports the overall results of FedRL cold start experiments in the four scenarios of initial incomplete information and data sparsity. We make the following observations:

- As the proportion of training set data decreases, the performance of FedRL in the four scenarios does not show a significant decline. This result demonstrates that FedRL exhibits good generalization when facing previously new users or scenarios, enabling rapid adaptation to new users. We attribute this to the federated architecture, which facilitates model information sharing among edge devices, enhancing the model's adaptability.
- In all scenarios, when the proportion is low (10% or 20%), the model's performance decline is more pronounced compared to higher proportions. The difficulty of capturing user preferences increases when the proportion is low,

but FedRL still exhibits robustness. We attribute this to FedRL's use of HNG and RLS, which enhance the model's ability to quickly adapt to new users. In this way, FedRL significantly mitigates the impact of η.

6 Conclusion

In this paper, we propose A Reinforcement Learning Federated Recommender System for Efficient Communication Using Reinforcement Selector and Hypernet Generator called FedRL, which utilizes the RLS and the HNG. It can employ the RLS for dynamic communication strategies and adaptive update client models using the HNG. This approach is designed to address recommender issues within scenarios of incomplete information and sparse data, while ensuring efficient communication. Through thorough experimentation on three extensively utilized datasets indicate that FedRL performs well in scenarios with incomplete information and sparse data. Additionally, FedRL demonstrates efficient communication efficiency owing to the ability of RLS to dynamically adjust communication between edge devices and third-party servers to adapt to different scenarios. Moreover, FedRL exhibits excellent generalization, signifying its capability to adapt to diverse scenarios while maintaining good performance across different environments. However, training RLS requires a large amount of empirical data. In the federated learning process, edge devices may join or leave the federated learning process at different time points, leading to the non-stationarity of FedRL. HNG, relying on user preference embeddings, may thus get trapped in local optimal solutions during training, making it difficult to find globally optimal parameter settings and thereby affecting the quality and effectiveness of the generated recommender network parameters.

Although our FedRL system is highly efficient and effective, there are still several directions that warrant exploration in the future. Firstly, we aim to incorporate multi-agent reinforcement learning into the client selection process within federated recommender system. This will enable the examination of strategy competition and adversarial learning among individual agents in competitive environments, aiming to enhance the learning process of personalized recommender strategies for improved personalization within the recommender system. Secondly, while FedRL has adopted a federated architecture to safeguard participant privacy, our future plan involve exploring more innovative methods. One potential approach is leveraging blockchain technology to further protect participant privacy. Exploring the utilization of blockchain in achieving data sharing and model aggregation can create a more secure and viable federated recommender system. Lastly, we are considering methods to enhance model communication efficiency further. Among these, a potential solution involves techniques such as model compression and knowledge distillation. Through these methods, we can reduce model size, thereby decreasing communication overhead and making communication more efficient.

Acknowledgments

This research was funded by the National Natural Science Foundation of China under grant number 61972182.

References

[1] Xiaowen Huang, Jitao Sang, Jian Yu, and Changsheng Xu. 2022. Learning to learn a cold-start sequential recommender. *ACM Transactions on Information Systems* 40, 2 (2022), 1-25.

[2] Wenmin Lin, Hui Leng, Ruihan Dou, Lianyong Qi, Zhigeng Pan, and Md. Arafatur Rahman. 2023. A federated collaborative recommendation model for privacy-preserving distributed recommender applications based on microservice framework. *Journal of Parallel and Distributed Computing* 174 (2023), 70-80.

[3] Rui Zhang, Xuesen Chu, Ruhui Ma, Meng Zhang, Liwei Lin, Honghao Gao, and Haibing Guan. 2022. OSTTD: Offloading of splittable tasks with topological dependence in multi-tier computing networks. *IEEE Journal on Selected Areas in Communications* 41, 2 (2022), 555-568.

[4] Xiaolong Xu, Wentao Liu, Yulan Zhang, Xuyun Zhang, Wanchun Dou, Lianyong Qi, and Md Zakirul Alam Bhuiyan. 2022. Psdf: Privacy-aware iov service deployment with federated learning in cloud-edge computing. *ACM Transactions on Intelligent Systems and Technology* 13, 5 (2022), 1-22.

[5] Feng Liang, Weike Pan, and Zhong Ming. 2021. Fedrec++: Lossless federated recommendation with explicit feedback. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35.

[6] Zhiwei Liu, Liangwei Yang, Ziwei Fan, Hao Peng, and Philip S. Yu. 2022. Federated social recommendation with graph neural network. *ACM Transactions on Intelligent Systems and Technology* 13, 4 (2022), 1-24.

[7] Muhammad Sarmad, Mishal Fatima, and Jawad Tayyub. 2022. Reducing Energy Consumption of Pressure Sensor Calibration Using Polynomial HyperNetworks with Fourier Features. In *Proceedings of the AAAI Conference on Artificial Intelligence,* Vol. 36.

[8] Zhongzhou Liu, Yuan Fang, and Min Wu. 2023. Dual-view preference learning for adaptive recommendation. *IEEE Transactions on Knowledge and Data Engineering*. (2023).

[9] Li Yin, Juan M. Perez-Rua, and Kevin J. Liang. 2022. Sylph: A hypernetwork framework for incremental few-shot object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9035-9045.

[10] David Ha, Andrew Dai, and Quoc V. Le. 2016. Hypernetworks. *arXiv preprint arXiv:1609.09106, (2016)*.

[11] Yuval Alaluf, Omer Tov, Ron Mokady, Rinon Gal, and Amit Bermano. 2022. Hyperstyle: Stylegan inversion with hypernetworks for real image editing, In *Proceedings of the IEEE/CVF conference on computer Vision and pattern recognition*, 18511-18521.

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

[13] Wentao Liu, Xiaolong Xu, Lianxiang Wu, Lianyong Qi, Alireza Jolfaei, Weiping Ding, and Mohammad R. Khosravi. 2022. Intrusion detection for maritime transportation systems with batch federated aggregation. *IEEE Transactions on Intelligent Transportation Systems* 24, 2 (2022), 2503-2514.

[14] Hongjian Shi, Weichu Zheng, Zifei Liu, Ruhui Ma, and Haibing Guan. 2023. Automatic Pipeline Parallelism: A Parallel Inference Framework for Deep Learning Applications in 6G Mobile Communication Systems. *IEEE Journal on Selected Areas in Communications*. (2023)

[15] Honghao Gao, Junsheng Xiao, Yuyu Yin, Tong Liu, and Jiangang Shi. 2022. A mutually supervised graph attention network for few-shot segmentation: the perspective of fully utilizing limited samples. *IEEE Transactions on neural networks and learning systems*. (2022).

[16] Songlei Wang, Yifeng Zheng, and Xiaohua Jia. 2023. SecGNN: Privacy-preserving graph neural network training and inference as a cloud service. *IEEE Transactions on Services Computing*. (2023).

[17] Jianqing Zhang , Yang Hua , Hao Wang , Tao Song , Zhengui Xue , Ruhui Ma , and Haibing Guan. 2023. FedCP: Separating Feature Information for Personalized Federated Learning via Conditional Policy, In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3249-3261.

[18] Jialiang Han , Yun Ma , Qiaozhu Mei , and Xuanzhe Liu. 2021. Deeprec: On-device deep learning for privacy-preserving sequential recommendation in mobile commerce, In *Proceedings of the Web Conference 2021*. 900-911.

[19] Muhammad Ammad-ud-din, Elena Ivannikova, Suleiman A. Khan, Were Oyomno, Qiang Fu, Kuan Eeik Tan, and Adrian Flanagan. 2019. Federated collaborative filtering for privacy-preserving personalized recommendation system. *arXiv preprint arXiv:1901.09888* (2019).

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

[21] Feng Liang, Weike Pan, and Zhong Ming. 2021. Fedrec++: Lossless federated recommendation with explicit feedback. In *Proceedings of the AAAI conference on artificial intelligence*. 4224-4231.

[22] Yicheng Di and Yuan Liu. 2023. MFPCDR: A Meta-Learning-Based Model for Federated Personalized Cross-Domain Recommendation. *Applied Sciences* 13(7), 4407.

[23] Omar Abdel Wahab , Gaith Rjoub , Jamal Bentahar , and Robin Cohen. 2022. Federated against the cold: A trust-based federated learning approach to counter the cold start problem in recommendation systems. *Information Sciences* 601, 189-206.

[24] 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.

[25] Steffen Rendle. 2012. Factorization machines with libfm. *ACM Transactions on Intelligent Systems and Technology* 3(3), 1-22.

[26] Heng-Tze Cheng , Levent Koc , Jeremiah Harmsen , Tal Shaked , Tushar Chandra , Hrishi Aradhye , Glen Anderson , Greg Corrado , Wei Chai , Mustafa Ispir , Rohan Anil , Zakaria Haque , Lichan Hong , Vihan Jain , Xiaobing Liu , and Hemal Shah. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*. 7-10.

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

[28] Yawei Li, Shuhang Gu, Kai Zhang, Luc Van Gool ,and Radu Timofte. 2020. Dhp: Differentiable meta pruning via hypernetworks. In *Computer Vision–ECCV 2020: 16th European Conference*. 608-624.

[29] Sriprabha Ramanarayanan, Arun Palla, Keerthi Ram, and Mohanasankar Sivaprakasam. 2023. Generalizing supervised deep learning mri reconstruction to multiple and unseen contrasts using meta-learning hypernetworks. *Applied Soft Computing* 146, 110633.

[30] Tan M. Dinh, Anh Tuan Tran, Rang Nguyen, and Binh-Son Hua. 2022. Hyperinverter: Improving stylegan inversion via hypernetwork. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11389-11398.

[31] Przemyslaw Spurek, Maciej Zieba, Jacek Tabor, and Tomasz Trzcinski. 2021. General hypernetwork framework for creating 3d point clouds. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44(12), 9995-10008.

[32] Etai Littwin, Tomer Galanti, Lior Wolf, and Greg Yang. 2020. On infinite-width hypernetworks. *Advances in neural information processing systems* 33, 13226-13237.

[33] Aviv Navon, Aviv Shamsian, Gal Chechik, and Ethan Fetaya. 2020. Learning the pareto front with hypernetworks. *arXiv preprint arXiv:2010.04104*

(2020).

[34] Yuval Nirkin, Lior Wolf, Tal Hassner. 2021. Hyperseg: Patch-wise hypernetwork for real-time semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4061-4070.

[35] Elad Sarafian, Shai Keynan, and Sarit Kraus. 2021. Recomposing the reinforcement learning building blocks with hypernetworks. In *International Conference on Machine Learning*. 9301-9312.

[36] Or Litany, Haggai Maron, David Acuna, Jan Kautz, Gal Chechik, and Sanja Fidler. 2022. Federated learning with heterogeneous architectures using graph hypernetworks. *arXiv preprint arXiv:2201.08459* (2022).

[37] Dominic Zhao, Seijin Kobayashi, João Sacramento, Johannes von Oswald. 2020. Meta-learning via hypernetworks. In 4*th Workshop on Meta-Learning at NeurIPS*.

[38] Aviv Shamsian, Aviv Navon, Ethan Fetaya, and Gal Chechik. 2021. Personalized federated learning using hypernetworks. In *International Conference on Machine Learning*. 9489-9502.

[39] Tomer Galanti and Lior Wolf. 2020. On the modularity of hypernetworks. *Advances in Neural Information Processing System* 33, 10409-10419.

[40] Xiaosong Ma, Jie Zhang, Song Guo, and Wenchao Xu. 2022. Layer-wised model aggregation for personalized federated learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10092-10101.

[41] Xiaofeng Fan, Yining Ma, Zhongxiang Dai, Wei Jing, Cheston Tan, and Bryan Kian Hsiang Low. 2021. Fault-tolerant federated reinforcement learning with theoretical guarantee. *Advances in Neural Information Processing Systems* 34, 1007-1021.

[42] Honghao Gao, Wanqiu Huang, Tong Liu, Yuyu Yin, and Youhuizi Li. 2022. Ppo2: Location privacy-oriented task offloading to edge computing using reinforcement learning for intelligent autonomous transport systems. *IEEE transactions on intelligent transportation systems*. (2022).

[43] Honghao Gao, Binyang Qiu, Ramón J. Durán Barroso, Walayat Hussain, Yueshen Xu, and Xinheng Wang. 2022. Tsmae: a novel anomaly detection approach for internet of things time series data using memory-augmented autoencoder. *IEEE Transactions on network science and engineering*. (2022).

[44] Yufeng Zhan, Peng Li, Zhihao Qu, Deze Zeng, and Song Guo. 2020. A learning-based incentive mechanism for federated learning. *IEEE Internet of Things Journal* 7(7), 6360-6368.

[45] Hongjian Shi, Hao Wang, Ruhui Ma, Yang Hua, Tao Song, Honghao Gao, and Haibing Guan. 2023. Robust searching-based gradient collaborative management in intelligent transportation system. *ACM Transactions on Multimedia Computing, Communications*. (2022).

[46] Yufeng Zhan, Peng Li, and Song Guo. 2020. Experience-driven computational resource allocation of federated learning by deep reinforcement learning. In *2020 IEEE International Parallel and Distributed Processing Symposium*. 234-243.

[47] Zhenni Li, Minrui Xu, Jiangtian Nie, Jiawen Kang, Wuhui Chen, and Shengli Xie. 2020. NOMA-enabled cooperative computation offloading for blockchain-empowered Internet of Things: A learning approach. *IEEE Internet of Things Journal* 8(4), 2364-2378.

[48] Xiaofei Wang, Ruibin Li, Chenyang Wang, Xiuhua Li, Tarik Taleb, and Victor C. M. Leung. 2020. Attention-weighted federated deep reinforcement learning for device-to-device assisted heterogeneous collaborative edge caching. *IEEE Journal on Selected Areas in Communications* 39(1), 154-169.

[49] Shuai Yu, Xu Chen, Zhi Zhou, Xiaowen Gong, and Di Wu. 2020. When deep reinforcement learning meets federated learning: Intelligent multitimescale resource management for multiaccess edge computing in 5G ultradense network. *IEEE Internet of Things Journal* 8(4), 2238-2251.

[50] Minrui Xu, Jialiang Peng, B. B. Gupta, Jiawen Kang, Zehui Xiong, Zhenni Li, and Ahmed A. Abd El-Latif. 2021. Multiagent federated reinforcement learning for Secure Incentive Mechanism in Intelligent Cyber–Physical Systems. *IEEE Internet of Things Journal* 9(22), 22095-22108.

[51] Hao Wang, Zakhary Kaplan, Di Niu, and Baochun Li. 2020. Optimizing federated learning on non-iid data with reinforcement learning. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. 1698-1707.

[52] Sai Qian Zhang, Jieyu Lin, and Qi Zhang. 2022. A multi-agent reinforcement learning approach for efficient client selection in federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 9091-9099.

[53] F. Maxwell Harper and Joseph A. Konstan. 2015. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems* 5(4), 1-19.

[54] Jiliang Tang, Huiji Gao, Xia Hu, and Huan Liu. 2013. Exploiting homophily effect for trust prediction. In *Proceedings of the sixth ACM international conference on Web search and data mining*. 53-62.

[55] Hao Ma, Dengyong Zhou, Chao Liu, Michael R. Lyu, and Irwin King. 2011. Recommender systems with social regularization. In *Proceedings of the fourth ACM international conference on Web search and data mining*. 287-296.

[56] Yan Zhang, Hongzhi Yin, Zi Huang, Xingzhong Du, Guowu Yang, and Defu Lian. 2018. Discrete deep learning for fast content-aware recommendation In *Proceedings of the eleventh ACM international conference on web search and data mining*. 717-726.

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

[58] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *IEEE international conference on data mining*. 197-206.

[59] Ilya Shenbin, Anton Alekseev, Elena Tutubalina, Valentin Malykh, Sergey I. Nikolenko. 2020. Recvae: A new variational autoencoder for top-n recommendations with implicit feedback. In *Proceedings of the 13th international conference on web search and data mining*. 528-536.

[60] Qiaomin Yi, Ning Yang, and Philip S. Yu. 2021. Dual adversarial variational embedding for robust recommendation. *IEEE Transactions on Knowledge and Data Engineering*. (2021).

[61] Qinyong Wang, Hongzhi Yin, Tong Chen, Junliang Yu, Alexander Zhou, and Xiangliang Zhang. 2022. Fast-adapting and privacy-preserving federated recommender system. *The VLDB Journal*, 1-20.

[62] Yujie Lin, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Dongxiao Yu, Jun Ma, Maarten de Rijke, Xiuzhen Cheng. 2020. Meta matrix factorization for federated rating predictions. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 981-990.

[63] Yujie Lin, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Dongxiao Yu, Jun Ma, Maarten de Rijke, Xiuzhen Cheng. 2020. LightFR: Lightweight federated recommendation with privacy-preserving matrix factorization. *ACM Transactions on Information Systems* 41(4), 1-28.

[64] Karan Singhal, Hakim Sidahmed, Zachary Garrett, Shanshan Wu, John Rush, and Sushant Prakash. 2021. Federated reconstruction: Partially local federated learning. *Advances in Neural Information Processing Systems* 34, 11220-11232.

[65] Chunxu Zhang, Guodong Long, Tianyi Zhou, Peng Yan, Zijian Zhang, Chengqi Zhang, and Bo Yang. 2023. Dual Personalization on Federated Recommendation. a*rXiv preprint arXiv:2301.08143* (2023).

[66] Chi Zhang, Rui Chen, Xiangyu Zhao, Qilong Han, Li Li. 2023. Denoising and Prompt-Tuning for Multi-Behavior Recommendation. In *Proceedings*

of the ACM Web Conference 2023. 1355-1363.

[67] Yuhao Yang, Chao Huang, Lianghao Xia, Yuxuan Liang, Yanwei Yu, and Chenliang Li. 2022. Multi-behavior hypergraph-enhanced transformer for sequential recommendation. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*. 2263-2274.

[68] Ye Tao, Ying Li, Su Zhang, Zhirong Hou, Zhonghai Wu. 2022. Revisiting graph based social recommendation: A distillation enhanced social graph network. In *Proceedings of the ACM Web Conference 2022*. 2830-2838.

[69] Shansan Gong and Kenny Q. Zhu. 2022. Positive, negative and neutral: Modeling implicit feedback in session-based news recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1185-1195.

[70] Pengqing Hu, Zhaohao Lin, Weike Pan, Qiang Yang, Xiaogang Peng, and Zhong Ming. 2023. Privacy-preserving graph convolution network for federated item recommendation. *Artificial Intelligence* 324, 103996.

[71] Jayant Vyas, Bhumika, Debasis Das, and Santanu Chaudhury. 2023. Federated learning based driver recommendation for next generation transportation system. *Expert Systems with Applications* 225, 119951.

[72] Zheng Li, Muhammad Bilal, Xiaolong Xu, Jielin Jiang, and Yan Cui. 2022. Federated Learning-Based Cross-Enterprise Recommendation With Graph Neural Networks. *IEEE Transactions on Industrial Informatics* 19(1), 673-682.

[73] Hilit Segev and Gal Chechik. 2023. Personalized Federated Learning for Medical Segmentation using Hypernetworks. In *International Conference on Learning Representations*.

[74] Rongyu Zhang, Yun Che, Chenrui Wu, Fangxin Wang, and Jiangchuan Liu. 2023. Optimizing Efficient Personalized Federated Learning with Hypernetworks at Edge. *IEEE Network* 37(4), 120-126.

[75] Peihua Mai and Yan Pang. 2023. Vertical Federated Graph Neural Network for Recommender System. *arXiv preprint arXiv:2303.05786* (2023).

[76] Mubashir Imran, Hongzhi Yin, Tong Chen, Quoc Viet Hung Nguyen, Alexander Zhou, and Kai Zheng. 2023. ReFRS: Resource-efficient federated recommender system for dynamic and diversified user preferences. *ACM Transactions on Information Systems* 41(3), 1-30.

[77] Shuai Zhang, Lina Yao, Aixin Sun, Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM computing surveys* 52(1), 1-38.

[78] Minmin Chen, Bo Chang, Can Xu, and Ed H. Chi. 2021. User response models to improve a reinforce recommender system. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 121-129.

[79] Yikun Xian, Tong Zhao, Jin Li, Jim Chan, Andrey Kan, Jun Ma, Xin Luna Dong, Christos Faloutsos, George Karypis, S. Muthukrishnan, and Yongfeng Zhang. 2021. Ex3: Explainable attribute-aware item-set recommendations. In *Proceedings of the 15th ACM Conference on Recommender Systems*. 484-494.

[80] Zhiyong Jie, Shuhong Chen, Junqiu Lai, Muhammad Arif , and Zongyuan He. 2022. Personalized federated recommendation system with historical parameter clustering. *Journal of Ambient Intelligence and Humanized Computing* 14(8), 10555-10565.

[81] Wei Yuan, Hongzhi Yi, Fangzhao Wu, Shijie Zhang, Tieke He, and Hao Wang. 2023. Federated unlearning for on-device recommendation. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*. 393-401.

[82] Liang Qu, Ningzhi Tang, Ruiqi Zheng, Quoc Viet Hung Nguyen, Zi Huang, Yuhui Shi, and Hongzhi Yin. 2023. Semi-decentralized federated ego graph learning for recommendation. In *Proceedings of the ACM Web Conference 2023*. 339-348.

[83] Chuan Sun, Xiuhua Li, Junhao Wen, Xiaofei Wang, Zhu Han, and Victor C. M. Leung. 2023. Federated deep reinforcement learning for recommendation-enabled edge caching in mobile edge-cloud computing networks. *IEEE Journal on Selected Areas in Communications* 41(3), 690-705.

[84] Zhijie Xie and Shenghui Song. 2023. FedKL: Tackling data heterogeneity in federated reinforcement learning by penalizing KL divergence. *IEEE Journal on Selected Areas in Communications* 41(4), 1227-1242.

[85] Prayag Tiwari, Abdullah Lakhan, Rutvij H. Jhaveri, and Tor-Morten Grønli. 2023. Consumer-centric internet of medical things for cyborg applications based on federated reinforcement learning. *IEEE Transactions on Consumer Electronics*. (2023).

[86] Xiaokang Zhou, Xuzhe Zheng, Xuesong Cui, Jiashuai Shi, Wei Liang, Zheng Yan, Laurence T. Yang, Shohei Shimizu, and Kevin I-Kai Wang. 2023. Digital twin enhanced federated reinforcement learning with lightweight knowledge distillation in mobile networks. *IEEE Journal on Selected Areas in Communications*. (2023).

[87] Sirui Chen, Yuan Wang, Zijing Wen, Zhiyu Li, Changshuo Zhang, Xiao Zhang, Quan Lin, Cheng Zhu, and Jun Xu. 2023. Controllable Multi-Objective Re-ranking with Policy Hypernetworks. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 3855-3864.

[88] Chenglei Shen, Xiao Zhang, Wei Wei, and Jun X. 2023. Hyperbandit: Contextual bandit with hypernewtork for time-varying user preferences in streaming recommendation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 2239-2248.

[89] Qi Liu, Zhilong Zhou, Gangwei Jiang, Tiezheng Ge, and Defu Lian. 2023. Deep task-specific bottom representation network for multi-task recommendation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*. 1637-1646.