A Federated Framework for LLM-based Recommendation

Jujia Zhao[1], **Wenjie Wang**[2*], **Chen Xu**[3*], **See-Kiong Ng**[4], **Tat-Seng Chua**[2]
[1]Leiden Institute of Advanced Computer Science, Leiden University
[2]NExT++ Research Center, National University of Singapore
[3]Gaoling School of Artificial Intelligence, Renmin University of China
[4]Institute of Data Science and School of Computing, National University of Singapore
{zhao.jujia.0913, wenjiewang96}@gmail.com
xc_chen@ruc.edu.cn, {seekiong, dcscts}@nus.edu.sg

Abstract

Large Language Models (LLMs) have empowered generative recommendation systems through fine-tuning user behavior data. However, utilizing the user data may pose significant privacy risks, potentially leading to ethical dilemmas and violations of data protection regulations. To address the privacy concerns, Federated Learning for Recommendation (Fed4Rec) has been identified as a promising solution. However, directly applying Fed4Rec in the LLM context introduces two challenges: 1) exacerbated client performance imbalance, which ultimately impacts the system's long-term effectiveness, and 2) substantial client resource costs, posing a high demand for clients' both computational and storage capability to locally train and infer LLMs.

To tackle these challenges, we propose a federated framework for LLM-based recommendation (shorted as FELLRec). Generally, FELLRec designs two key strategies. 1) Dynamic balance strategy, which designs dynamic parameter aggregation and learning speed for different clients, aiming to ensure balanced performance across clients. 2) Flexible storage strategy, which selectively retains certain sensitive LLM layers on the client side, while offloading other layers to the server, aiming to preserve privacy while saving resources. Experiment results show that FELLRec can achieve a more balanced client performance and improved overall performance in a computational and storage-efficient way while safeguarding user privacy well.

1 Introduction

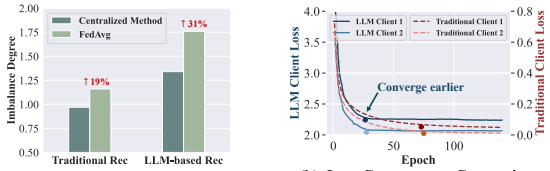
Large Language Models (LLMs) with advanced contextual understanding abilities have demonstrated potential in building generative recommendation systems (Rajput et al., 2023; Gao et al., 2023). Fine-tuning LLMs with user behavior data is essential for learning user preferences (Bao et al., 2023a; Li et al., 2023), however, it will face serious privacy leakage risks like in the traditional recommender models. The unintended disclosure of sensitive user data could cause ethical issues and infringe upon data protection laws such as the General Data Protection Regulation in the European Union (Hoofnagle et al., 2019). Therefore, ensuring the security and privacy of recommendation data during the LLM fine-tuning process is crucial.

To address the data privacy concerns, Federated Learning for Recommendation (Fed4Rec) emerges as a promising solution (Muhammad et al., 2020; Sun et al., 2022). Fed4Rec requires clients (*e.g.,* user devices and platforms with a group of users) to conduct local training using the client's data, and then exchange non-sensitive intermediate parameters such as model parameters and gradients. This approach protects sensitive user behavior data by keeping them on the client side without the need for sharing with others. In General, Fed4Rec mainly employs two frameworks: 1) Peer-Peer Framework (Yang et al., 2022; An et al., 2024), which makes every client broadcast the updated parameters to other clients directly within the peer-to-peer network. However, this framework faces limitations in LLM-based recommendation scenarios, primarily due to the high communication costs incurred by the large number of LLM parameters. 2) Client-Server Framework (Zhang et al., 2023a,b), which transmits the updated parameters of the clients to a central server for aggregation. Previous works (Sun et al., 2022; Yin et al., 2024) have demonstrated that the client-server framework is more efficient in terms of communication overhead, making it ideal for LLM-based recommendations.

However, adapting the client-server framework to LLM-based recommendation presents two challenges: 1) **Exacerbated Client Performance Imbalance**: Based on our empirical analysis in Figure 1(a), it is evident that directly applying the

*Corresponding author.

Findings of the Association for Computational Linguistics:
NAACL 2025, pages 2852–2865



(a) Client Performance Imbalance Comparison (b) Loss Convergence Comparison

Figure 1: (a) illustrates the exacerbated client performance imbalance when applying a classical client-server method (FedAvg (McMahan et al., 2017)) to LLM-based recommender models (BIGRec) compared with traditional recommender models (MF). (b) shows the convergence rate of two selected clients when applying FedAvg to LLM-based and traditional models. The observations are on Games.

client-server framework to LLM-based recommendation models leads to a more significant client performance imbalance compared to traditional models. This exacerbated imbalance may cause less accurate and equitable recommendations for specific clients, ultimately impacting the system's long-term effectiveness and user satisfaction (Xu et al., 2023a; Burke et al., 2018). This exacerbated imbalance potentially stems from the accelerated training convergence among clients, as depicted in Figure 1(b), possibly due to the fast adaptation capabilities of LLMs (Bao et al., 2023a,b). 2) **Substantial Client Resource Costs**: The client-server framework necessitates that each client possesses the capability to locally train and infer LLMs. However, the extensive computational and storage resources required by LLMs pose a substantial challenge for individual clients in meeting these demands (Chen et al., 2023; Fan et al., 2023).

To tackle the issues of exacerbated performance imbalance and substantial resource costs, we refine the client-server framework with two strategies: 1) **Dynamic Balance Strategy**. To mitigate the performance imbalance among clients, we introduce a dynamic balance strategy: it involves designing dynamic parameter aggregation and learning speed for each client to ensure relatively equitable performance across the board. 2) **Flexible Storage Strategy**: To reduce client costs, we propose a flexible storage strategy for the client model. Intuitively, this strategy selectively allocates some LLM layers, especially those capable of extracting sensitive user data, on the client side, while situating other non-sensitive layers on the server to save cost.

In light of these, we propose a **Fe**derated Framework for **LL**M-based Recommendation (FELLRec). 1) FELLRec adapts dynamic balance strategies for different clients. Specifically, FELLRec preserves personalized parameters on each client (*e.g.,* Low-Rank Adaption (LoRA) (Hu et al., 2021)) and employs a dynamic parameter aggregation method based on attention mechanisms. Meanwhile, FELLRec devises dynamic learning speed by proposing a Curriculum Heating learning method (Chen et al., 2021) based on client loss, which helps client undergoes a gradual prewarming phase to familiarize themselves with their own data distribution. 2) FELLRec adopts the flexible storage strategy to deploy those input and output layers on the client side to ensure the protection of all sensitive information (see detailed analysis in Section 4.3). Empowered with the two strategies, FELLRec can safeguard data privacy for LLM-based recommendations in a more balanced and efficient way. We instantiate FELLRec on two LLM backend models and conduct extensive experiments on three datasets, validating its effectiveness and efficiency.

Main Contributions: (1) We introduce a privacy-preserving task for fine-tuning LLM-based recommendation models, where we identify the challenges of directly adopting Fed4Rec: exacerbated client performance imbalance and substantial client resource costs. (2) We propose a federated framework for LLM-based recommendation called FELLRec, which addresses the two challenges well while preserving data privacy. (3) Experiments across three public datasets under various settings, confirming its efficacy and efficiency.

2 Preliminary

2.1 LLM-based Recommendation

Let $\mathcal{U}$, $\mathcal{I}$ be the user set and item set, for a given user $u \in \mathcal{U}$, the LLM-based recommender $f(\boldsymbol{\mathcal{P}})$ will utilize the user's historical interactions H_u to generate a ranking list $L_K(H_u) \subset \mathcal{I}$ as the recommendation for user u, where K is the item numbers in a ranking list and $\boldsymbol{\mathcal{P}}$ is the parameter set of LLMs. H_u is the user u's browsing history: $H_u = [i_1, i_2, \cdots, i_N]$, where $i_n \in \mathcal{I}$ is the $n-$th item in the interaction history (typically in a natural language form), and N is the history length.

2.2 Client-Server Framework under Fed4Rec

Let $\mathcal{C}$ be the client set, where each client $c \in \mathcal{C}$ could be a user u or a group of users from a specific platform. Each client c, equipped with a

Our code and data are released at https://github.com/Polaris-JZ/FELLRec.

model parameter $\boldsymbol{\mathcal{P}}_c$, has a local dataset $\boldsymbol{\mathcal{D}}_c = \{(H_u, y), \forall u \in c\}$, which includes the users' interaction history H_u and the label y (usually the next-interacted item) for training. Within the client-server framework, the most classic approach is FedAvg (McMahan et al., 2017). Specifically, at each training epoch, FedAvg utilizes a central server to aggregate parameters from various clients to generate unified updated parameters for every client. Formally, at each epoch, each client is required to update its parameter based on its local dataset: $\boldsymbol{\mathcal{P}}_c = \arg\min_{\boldsymbol{\mathcal{P}}_c} \sum_{(H_u,y)\in\boldsymbol{\mathcal{D}}_c} l(f(H_u; \boldsymbol{\mathcal{P}}_c), y), \forall c \in \mathcal{C}$, where $l(\cdot)$ is the loss function of recommendation. Subsequently, the central server will aggregate parameters from all clients, and send the unified aggregated parameters back to every client: $\boldsymbol{\mathcal{P}}_c = \frac{1}{n}\sum_{c'\in\mathcal{C}} \frac{|\boldsymbol{\mathcal{D}}_{c'}|}{|\boldsymbol{\mathcal{D}}|}\boldsymbol{\mathcal{P}}_{c'}, \forall c \in \mathcal{C}$. FedAvg ensures privacy by obviating the necessity to transmit original data to the server, concentrating instead on the exchange of model parameters. However, directly applying FedAvg to LLM-based recommendations will meet the exacerbated client performance imbalance and substantial client resource cost.

3 FELLRec

In response to exacerbated client performance imbalance and significant client resource costs, we introduce a Federated Framework for LLM-based Recommendation (FELLRec), which enhances data privacy in LLM-based recommendation systems both equitably and efficiently. FELLRec encompasses two strategies: 1) Dynamic balance strategy, which designs dynamic parameter aggregation and learning speeds to ensure relatively balanced performance across clients. 2) Flexible storage strategy, which enables the flexible storage of LLM layers to conserve resources. The architecture of FELLRec is depicted in Figure 2.

3.1 Dynamic Balance Strategy

As illustrated in Section 1, directly applying the client-server framework in LLM-based recommendation exacerbates the performance imbalance across clients. This imbalance may lead to less equitable recommendations for specific clients, thereby detrimentally affecting the system's overall effectiveness and diminishing user satisfaction.

The imbalance could be potentially attributed to two primary factors: 1) the diverse data distribution among clients, which may lead to conflicting optimization objectives among clients, thus possibly sacrificing the performance of specific clients. 2) The varied learning difficulty levels among clients, where those facing greater challenges may exhibit relatively poorer performance.

To address these issues, FELLRec first ensures client personalization by maintaining client-specific parameters for each client, including two kinds: 1) LoRA (Hu et al., 2021), and 2) either a part or the entirety of LLM's own parameters. To economize on client resources, the remaining parameters are fixed. Our analysis primarily utilizes LoRA as an illustrative example, as the same principles apply to other methods.

Specifically, for client c, the model parameters are denoted as $\boldsymbol{\mathcal{P}}_c$ with LoRA $\boldsymbol{\mathcal{R}}_c$, where LoRA is client-specific parameters and $\boldsymbol{\mathcal{P}}_c$ is the fixed original LLM model parameters. Subsequently, FELLRec incorporates a dynamic balance strategy, which involves designing dynamic parameter aggregation and learning speeds for each client, addressing two key factors of imbalance respectively.

An intuitive idea of our method is:

$$\boldsymbol{\mathcal{R}}_c = \frac{\sum_{c'\in\mathcal{C}} d_{c,c'}\boldsymbol{\mathcal{R}}_{c'}}{\sum_{c'\in\mathcal{C}} d_{c,c'}} \tag{1}$$

where $d_{c,c'}$ is the dynamic aggregation weight and it can be divided into two parts: $d_{c,c'} = w_c \cdot s_{c,c'}$, where $s_{c,c'}$ is the attention-based aggregation weight corresponding to the Sub-section 3.1.1 and w_c is the learning difficulty weight illustrated at the Sub-section 3.1.2.

3.1.1 Dynamic Parameter Aggregation

Given the variability in data distribution among clients, the optimization objectives for them may diverge, potentially leading to conflicts when trying to optimize a global model. Such conflicts may inadvertently sacrifice the performance of specific clients, which causes imbalance.

Given this, FELLRec incorporates an attention-based parameter aggregation method. This method customizes the aggregation process of each client according to their unique data distribution, aiming to mitigate performance imbalances without compromising the performance of specific clients. Intuitively, the client model prioritizes learning from clients with similar data distributions while reducing the influence of those deemed non-relevant. The prioritization mechanism involves aggregating the client model parameters based on the cosine similarity between the parameters of the current client and those of other clients. Specifically, for

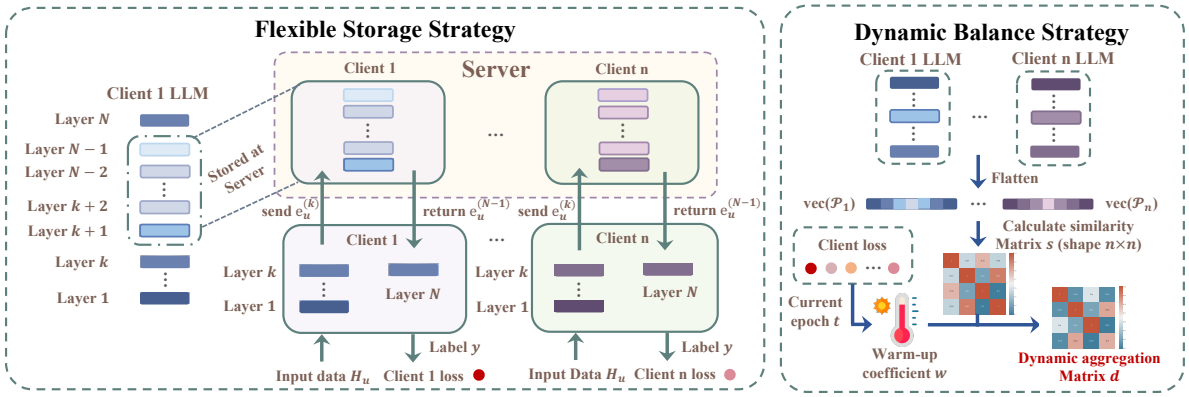


Figure 2: FELLRec Structure. The left part is the flexible storage strategy which offloads non-sensitive LLM layers to the server to save resources. The right part is the dynamic balance strategy which ensures relatively balanced performance across clients.

client c, the aggregation formula is:

$$\boldsymbol{\mathcal{R}}_c = \frac{\sum_{c' \in \mathcal{C}} s_{c,c'} \boldsymbol{\mathcal{R}}_{c'}}{\sum_{c' \in \mathcal{C}} s_{c,c'}}, \quad \text{where} \tag{2}$$

$$s_{c,c'} = \frac{\text{vec}(\boldsymbol{\mathcal{R}}_c)^\top \text{vec}(\boldsymbol{\mathcal{R}}_{c'})}{\|\text{vec}(\boldsymbol{\mathcal{R}}_c)\|_2 \|\text{vec}(\boldsymbol{\mathcal{R}}_{c'})\|_2}. \tag{3}$$

$\|\cdot\|_2$ denotes the ℓ_2 norm, $\boldsymbol{\mathcal{R}}_c$ represents the client-specific LoRA parameter of client c, $\text{vec}(\boldsymbol{\mathcal{R}}_c)$ represents flattened one-dimension client-specific LoRA parameter of client c, and $s_{c,c'}$ is attention-based aggregation weight, which is cosine similarity between $\text{vec}(\boldsymbol{\mathcal{R}}_c)$ and $\text{vec}(\boldsymbol{\mathcal{R}}_{c'})$.

Through dynamic parameter aggregation, FELLRec ensures more balanced performance across clients by customizing the aggregation process of each client based on its specific data distribution.

3.1.2 Dynamic Learning Speed

Given the varied heterogeneity within client datasets, clients encounter different learning difficulties during training (Yang et al., 2023). Consequently, the learning status of different clients (*e.g.,* ongoing learning, convergence or overfitting) can vary significantly. If a client has not adequately learned from its own data, excessively aggregating parameters from other clients may detrimentally affect its performance, potentially leading to performance imbalances across clients.

In response to this challenge, we develop a client-specific dynamic learning speed mechanism. This mechanism dynamically adjusts the extent of learning from other clients according to the client's current learning status, thereby personalizing the client's learning process. FELLRec assesses a client's learning status via its local loss, which serves as a gauge of the client's learning difficulty, and adjusts the extent of learning from peers accordingly. Based on this, FELLRec introduces a Curriculum Heating learning method (Chen et al., 2021), which is adapted based on client loss. Intuitively, clients experiencing higher losses undergo a gradual pre-warming phase, allowing them to acclimate to their data distribution, whereas clients with lower losses engage in a rapid convergence, enhancing training efficiency. Specifically, for client c, warm-up coefficient is:

$$w_c = \tanh\left(\frac{\alpha}{\left[\exp(\mathcal{L}_c)/\sum_{i=1}^{N}\exp(\mathcal{L}_i)\right]^{t/\beta}}\right), \tag{4}$$

where α is the speed-related warm-up factor, influencing the warm-up's overall pace; β is the time-related warm-up factor, affecting the temporal impact on warm-up speed. In essence, a higher α or a lower β accelerates warm-up for clients. w_c is posed on the similarity score with other clients to control the learning speed: $d_{c,c'} = w_c s_{c,c'}, \quad \forall c' \in \mathcal{C}, c' \neq c$, where $d_{c,c'}$ is the final dynamic aggregation weight. This approach, through the application of the warm-up coefficient, dynamically adjusts a client's learning pace based on its current learning status, providing a tailored learning speed for each client and mitigating performance imbalances across the clients.

3.2 Flexible Storage Strategy

In LLM-based recommendation systems, training and inference processes demand significant resource investment. Recognizing that not all clients have the capacity for the storage and computational demands of an LLM model, FELLRec introduces a flexible storage strategy aimed at reducing resource expenditure for clients.

FELLRec retains specific subsets of layers on the client side, particularly those closer to the in-

The practical application of Apple Inc. demonstrates the

put and output layers, due to their processing of sensitive data. The rest of the layers are hosted on the server side to save resources. Speicfically, for client c, the client model parameters are denoted as $\boldsymbol{\mathcal{P}}_c$ with LoRA $\boldsymbol{\mathcal{R}}_c$. Both of this two parts $\boldsymbol{\mathcal{P}}_c$ and $\boldsymbol{\mathcal{R}}_c$ can be divided into N layers: $\{\boldsymbol{\mathcal{P}}_c^{(i)}\}_{i=1}^N$, $\{\boldsymbol{\mathcal{R}}_c^{(i)}\}_{i=1}^N$, respectively, where N is the total number of LLM layers. Therefore, we combine them as $\boldsymbol{\mathcal{T}}_c$ for simplicity, where $\boldsymbol{\mathcal{T}}_c = \{\boldsymbol{\mathcal{P}}_c^{(i)}, \boldsymbol{\mathcal{R}}_c^{(i)}\}_{i=1}^N$ Based on this, The layer retained on the client side are $\{\boldsymbol{\mathcal{T}}_c^{(i)}\}_{i=1}^k$ and $\boldsymbol{\mathcal{T}}_c^{(N)}$, where k represents the layer-allocation hyper-parameter. Conversely, the layers stored on the server side are $\{\boldsymbol{\mathcal{T}}_c^{(i)}\}_{i=k+1}^{N-1}$.

During each training round, the client sends input data H_u to its preserved input layers, which then forwards the output embedding $\boldsymbol{e}_u^{(k)} = g(H_u, \{\boldsymbol{\mathcal{T}}_c^{(i)}\}_{i=1}^k)$ to the server for further processing, where $g(\cdot)$ commonly is the attention layer with feed-forward layer in the LLM scenario. The server processes this embedding and returns the output $\boldsymbol{e}_u^{(N-1)} = g(\boldsymbol{e}_u^{(k)}, \{\boldsymbol{\mathcal{T}}_c^{(i)}\}_{i=k+1}^{N-1})$ to the client to produce the final output embedding $\boldsymbol{e}_u^N = g(\boldsymbol{e}_u^{(N-1)}, \boldsymbol{\mathcal{T}}_c^{(N)})$. Subsequently, the client calculates the loss using the preserved label on the client side. Formally, the forward process of FELLRec is described as $\boldsymbol{\mathcal{T}_c} = \{\boldsymbol{\mathcal{T}}_c^{(i)}\}_{i=1}^k \circ \{\boldsymbol{\mathcal{T}}_c^{(i)}\}_{i=k+1}^{N-1} \circ \boldsymbol{\mathcal{T}}_c^{(N)}$, where $\circ$ represents operation composition, with the output of the function on the right being used as the input to the function on the left. Following this, the backward process begins, with gradients propagated in reverse: from the client to the server and then back to the client.

When client and server configurations are consistent, FELLRec's performance is unaffected by the parameter k since offloading layers to the server doesn't alter the training mechanics—only the storage location of model parameters changes. Specifically, both the forward and backward propagation processes proceed identically to scenarios where no layers are offloaded to the server.

This strategy significantly reduces client resource costs during both training and inference, as shown in Table 3. It is noteworthy that the determination of the number of layers to preserve is adaptable, enabling control over client costs. However, despite our method's efforts to protect data privacy, there may be malicious behavior from the server side aimed at attacking the model to access user privacy data. Our subsequent experiments indicate that retaining more layers on the server side increases the vulnerability to attacks (as detailed in Section 4.3), where we analyze the trade-off between the risk of attacks and the costs.

feasibility of deploying LLMs, such as those with 3 billion or 7 billion parameters, on the client side. (Inc., 2024)

Algorithm 1 FELLRec Training Phase

Require: The client set $\mathcal{C}$, item set $\mathcal{I}$, epoch number T, local round number R, warm-up parameter α, β, personalized parameters $\mathcal{R}_c$ and local data $\mathcal{D}_c = \{H_u, y\}, \forall c \in \mathcal{C}$.

Ensure: Fine-tuned personalized parameters $\mathcal{R}_c, \forall c \in \mathcal{C}$.

Initialize client model $\mathcal{R}_c, c \in \mathcal{C}$
for all each epoch $t = 1, 2, \cdots, T$ **do**
  Initialize $\mathcal{L}_c = 0$
  // Client Local Training
  for all each client $c \in \boldsymbol{\mathcal{C}}$ in parallel **do**
    for all each round $r = 1, 2, \cdots, R$ **do**
      $l_c(\boldsymbol{\mathcal{R}_c}) = \sum_{(H_u, y) \in \mathcal{D}_c} l(f(H_u; \mathcal{R}_c), y)$
      $\mathcal{L}_c = \mathcal{L}_c + l_c$
      $\mathcal{R}_c = \arg\min_{\boldsymbol{\mathcal{R}_c}} l_c(\mathcal{R}_c)$
  Upload $\{\boldsymbol{\mathcal{R}_c^{(i)}}\}_{i=1}^k, \boldsymbol{\mathcal{R}_c^{(N)}}, \forall c$ to server
  // Aggregate param. for clients
  $w_c = \tanh(\frac{\alpha}{\left[\exp(\mathcal{L}_c)/\sum_{i=1}^N \exp(\mathcal{L}_i)\right]^{t/\beta}})$
  $s_{c,c'} = \frac{\mathrm{vec}(\boldsymbol{\mathcal{R}}_c)^\top \mathrm{vec}(\boldsymbol{\mathcal{R}}_{c'})}{\|\mathrm{vec}(\boldsymbol{\mathcal{R}}_c)\|_2 \|\mathrm{vec}(\boldsymbol{\mathcal{R}}_{c'}\|_2)}, \forall c, c' \in \mathcal{C}$
  $d_{c,c'} = w_c s_{c,c'}$
  $\boldsymbol{\mathcal{R}}_c = \frac{\sum_{c' \in \mathcal{C}} d_{c,c'} \boldsymbol{\mathcal{R}}_{c'}}{\sum_{c' \in \mathcal{C}} d_{c,c'}}, \forall c \in \mathcal{C}$
  Send $\{\boldsymbol{\mathcal{R}_c^{(i)}}\}_{i=1}^k, \boldsymbol{\mathcal{R}_c^{(N)}}, \forall c \in \mathcal{C}$ back to clients

3.3 FELLRec Framework

3.3.1 Training

In the training phase, FELLRec trains personalized parameter $\boldsymbol{\mathcal{R}}_c$ for each client c without sharing their data. Specifically, at each epoch t, FELLRec first conducts client local training and then aggregates parameters of all clients to update their personalized parameter $\boldsymbol{\mathcal{R}}_c$.

Specifically, at the client local training phase, each client updates their parameters $\boldsymbol{\mathcal{R}}_c$ utilizing their respective local datasets $\boldsymbol{\mathcal{D}}_c$. During this phase, the client model is not entirely stored on the client side. Instead, parts of the model are stored on the server side, as dictated by the flexible storage strategy, to reduce the resource costs associated with training LLMs (see detailed analysis Section 3.2). Subsequently, each client uploads their client-preserved parameters to the server for aggre-

Algorithm 2 FELLRec Inference Phase

Require: The client set $\mathcal{C}$, item set $\mathcal{I}$, ranking size K, parameters of each client $\boldsymbol{\mathcal{T}}_c, \forall c \in \mathcal{C}$, the user u

Ensure: Ranking list $L_K(u)$

// Offline Storage
for all client $c \in \mathcal{C}$ **do**
    Get item embeddings $\boldsymbol{e}_c(i) = f(i, \{\boldsymbol{\mathcal{T}}_c^{(i)}\}_{i=1}^{N}), \forall i$
User u arrives in FELLRec;
Finding u corresponds to the client c;
// Client c executes
$\boldsymbol{e}_u^{(k)} = g(H_u, \{\boldsymbol{\mathcal{T}}_c^{(i)}\}_{i=1}^{k})$
Upload $e_u^{(k)}$ to Server;
// Server executes
$\boldsymbol{e}_u^{(N-1)} = g(\boldsymbol{e}_u^{(k)}, \{\boldsymbol{\mathcal{T}}_c^{(i)}\}_{i=k+1}^{N-1})$
Upload $\boldsymbol{e}_u^{(N-1)}$ to Client;
// Client c executes
Get output embedding $\boldsymbol{e}_u^{(N)} = g(\boldsymbol{e}_u^{(N-1)}, \{\boldsymbol{\mathcal{T}}_c^{(N)}\})$
// Ranking Step
$L_K(u) = \arg\min_{S \subset \mathcal{I}, |S|=K} \sum_{i \in S} \text{distance}(\boldsymbol{e}_c(i), \boldsymbol{e}_u^{(N)})$

gation, making use of the parameters from other clients to assist the update process.

In the aggregate phase, each client gets their dynamic aggregation weight $d_{c,c'}, \forall c' \in \mathcal{C}$ through dynamic parameter aggregation and dynamic learning speed. Subsequently, they get aggregated personalized parameters $\boldsymbol{\mathcal{R}}_c$ based on their specific aggregation weight and then send client-preserved parameters back to clients. The training algorithm of FELLRec is provided in Algorithm 1.

3.3.2 Inference

In the inference phase, for any given client c, FELLRec utilizes the updated LoRA parameters $\boldsymbol{\mathcal{R}}_c$ and fixed parameters $\boldsymbol{\mathcal{P}}_c$ to form the complete parameters $\boldsymbol{\mathcal{T}}_c$, and then get ranking list L_K as recommendation for user u belongs to this client.

Specifically, the inference phase is divided into four phases: 1) Client c independently stores the embeddings $\boldsymbol{e}_c(i)$ for all items from the item corpus $\mathcal{I}$, in preparation for the ranking step. 2) Client c gets the hidden embedding at $k-$th layer of LLM through: $\boldsymbol{e}_u^{(k)} = g(H_u, \{\boldsymbol{\mathcal{T}}_c^{(i)}\}_{i=1}^{k})$; 3) Then the server receives the uploaded $\boldsymbol{e}_u^{(k)}$ and continue to compute the hidden embedding $\boldsymbol{e}^{(N-1)}$ at $(N-1)-$th layer of LLM through $\boldsymbol{e}_u^{(N-1)} = g(\boldsymbol{e}_u^{(k)}, \{\boldsymbol{\mathcal{T}}_c^{(i)}\}_{i=k+1}^{N-1})$; 4) Finally, client c directly computes the distance (*e.g.,* cosine similarity(Bao et al., 2023a) or L2 distance (Li et al., 2023)) between the generated embedding $\boldsymbol{e}_u^{(N)}$ and the item embeddings $\boldsymbol{e}_c(i)$ from item corpus, and get the final ranking list through: $L_K(u) = \arg\min_{S \subset \mathcal{I}, |S|=K} \sum_{i \in S} \text{distance}(\boldsymbol{e}_c(i), \boldsymbol{e}_u^{(N)})$.

This approach preserves sensitive user data on the client side during inference, thus enhancing data privacy. Additionally, by offloading portions of the computation to the server, FELLRec reduces the computational load on clients and minimizes their hardware requirements. The inference algorithm of FELLRec is provided in Algorithm 2.

4 Experiments

In this section, we conduct a comprehensive experimental study to analyze the performance of FELLRec and the impact of different components (*e.g.,* dynamic balance strategy and flexible storage strategy) within it.

4.1 Experimental Settings

4.1.1 Datasets and Settings

We assess the effectiveness of FELLRec on three popular benchmark datasets. 1) **Games** is from the Amazon review datasets, which covers interactions between users and video games with rich textual features such as game titles and categories. 2) **MicroLens** is a newly released short video recommendation dataset. Each short video contains raw modal information such as title, cover image, audio, and video information. 3) **Book** is also derived from Amazon review datasets, containing users' interactions with extensive books, encompassing a broad spectrum of genres and subjects. The datasets' statistics are detailed in Table 5. For all three datasets, we organize user-item interactions chronologically based on timestamps and divide the data into training, validation, and testing sets in an 8:1:1 ratio.

Within the context of LLM-based recommendations, we explore two distinct fine-tuning approaches: 1) **Few-shot fine-tuning** fine-tunes LLMs using a limited number of examples, *e.g.,* 1024-shot. 2) **Full fine-tuning** utilizes all samples to fine-tune LLMs.

For the evaluation, we adopt full-ranking protocol (He et al., 2020) and evaluate using Recall@K and NDCG@K, where $K = 10$ or 20.

4.1.2 Baselines

We compare FELLRec against competitive baselines. First, we select two superior backend

https://nijianmo.github.io/amazon/index.html.
https://github.com/westlake-repl/MicroLens.

Table 1: Overall performance of FELLRec and other baselines in the LLM-based Recommendation scenario. Bold signifies the best performance among the privacy-preserving methods under the same backend models. * denotes statistically significant improvements of FELLRec over the second-best privacy-preserving methods under the same backend models, according to the t-tests with a significance level of p <0.01.

Method	Games				Microlens				Book			
	R@10	R@20	N@10	N@20	R@10	R@20	N@10	N@20	R@10	R@20	N@10	N@20
BIGRec	0.0194	0.0316	0.0127	0.0164	0.0089	0.0132	0.0050	0.0062	0.0079	0.0097	0.0126	0.0116
+FedAvg	0.0145	0.0257	0.0093	0.0126	0.0021	0.0039	0.0012	0.0017	0.0081	0.0097	0.0119	0.0112
+FedProx	0.0143	0.0255	0.0090	0.0123	0.0033	0.0051	0.0032	0.0040	0.0081	0.0096	0.0120	0.0112
+Ditto	0.0147	0.0260	0.0091	0.0126	0.0040	0.0063	0.0041	0.0045	0.0077	0.0091	0.0113	0.0106
+RoLoRA	0.0128	0.0231	0.0079	0.0106	0.0019	0.0037	0.0013	0.0019	0.0052	0.0075	0.0101	0.0098
+Ours	**0.0158***	**0.0274***	**0.0104***	**0.0139***	**0.0088***	**0.0128***	**0.0051***	**0.0062***	**0.0085***	**0.0102***	**0.0124***	**0.0116***
RecFormer	0.0193	0.0360	0.0117	0.0169	0.0190	0.0369	0.0104	0.0155	0.0318	0.0512	0.0333	0.0380
+FedAvg	0.0149	0.0262	0.0089	0.0124	0.0086	0.0192	0.0045	0.0074	0.0078	0.0123	0.0085	0.0097
+FedProx	0.0150	0.0266	0.0086	0.0121	0.0086	0.0166	0.0041	0.0064	0.0071	0.0061	0.0083	0.0133
+Ditto	0.0162	0.0273	0.0091	0.0138	0.0091	0.0172	0.0046	0.0065	0.0102	0.0131	0.0107	0.0159
+RoLoRA	0.0132	0.0257	0.0081	0.0118	0.0084	0.0187	0.0029	0.0045	0.0071	0.0115	0.0079	0.0095
+Ours	**0.0215***	**0.0373***	**0.0122***	**0.0170***	**0.0141***	**0.0245***	**0.0065***	**0.0094***	**0.0274***	**0.0411***	**0.0275***	**0.0301***

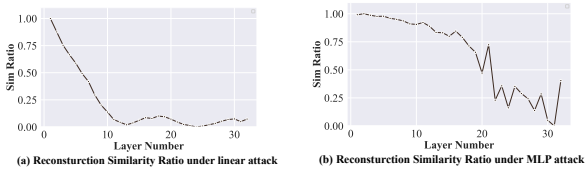
Table 2: Overall performance of FELLRec and other traditional recommendation baselines. Bold signifies the best performance among all methods. Underlined values indicate the second best. * denotes statistically significant improvements of the best method over the second-best, according to the t-tests with a significance level of p <0.01.

Method		Games				Microlens				Book			
		R@10	R@20	N@10	N@20	R@10	R@20	N@10	N@20	R@10	R@20	N@10	N@20
Centralized	**MF**	0.0101	0.0164	0.0070	0.0090	0.0044	0.0063	0.0026	0.0032	0.0050	0.0089	0.0060	0.0071
	LightGCN	0.0153	0.0234	0.0101	0.0127	0.0078	0.0116	0.0044	0.0055	0.0065	0.0120	0.0078	0.0093
Federated	**FedMF**	0.0065	0.0108	0.0044	0.0058	0.0029	0.0045	0.0021	0.0027	0.0050	0.0070	0.0034	0.0041
	LightFR	0.0088	0.0139	0.0051	0.0069	0.0041	0.0055	0.0024	0.0044	0.0048	0.0079	0.0049	0.0061
	FedPerGNN	0.0145	0.0229	0.0093	0.0121	0.0043	0.0060	0.0024	0.0029	0.0062	0.0112	0.0075	0.0089
	BIGRec+Ours	0.0158	0.0274	0.0104	0.0139	0.0088	0.0128	0.0051	0.0062	0.0085	0.0102	0.0124	0.0116
	RecFormer+Ours	**0.0215***	**0.0373***	**0.0122***	**0.0170***	**0.0141***	**0.0245***	**0.0065***	**0.0094***	**0.0274***	**0.0411***	**0.0275***	**0.0301***

LLMs: 1) **BIGRec** (Bao et al., 2023a). 2) **RecFormer** (Li et al., 2023). Given the absence of LLM-based privacy-preserving recommendation method in existing literature, we incorporate two well-established federated learning algorithms that can be deployed on LLM: 3) **FedAvg** (McMahan et al., 2017). 4) **FedProx** (Li et al., 2020). 5) **Ditto** (Li et al., 2021). 6) **RoLoRA** (Chen et al., 2024). Additionally, our comparison also includes baselines from traditional recommendation methods. Specifically, we select **MF** (Gantner et al., 2011) and **LightGCN** (He et al., 2020) as the centralized-based method, along with their federated counterparts: **FedMF** (Chai et al., 2020), **LightFR** (Zhang et al., 2023b), and **FedPerGNN** (Wu et al., 2022). Details of baselines and implementation are shown in Appendix A.1 and A.2.

4.2 Overall Performance

We compare FELLRec with other baselines, shown in Table 1 and Table 2. The result indicates that: 1) FELLRec outperforms other privacy-preserving methods on all datasets and achieves performance on par with centralized LLM-based methods. This efficacy is largely due to the dynamic balance strategy, which offers dynamic parameter aggregation and learning speeds. 2) FedAvg and FedProx performance fluctuates due to their inability to robustly adapt to varied data distributions across clients and the heterogeneity within clients (see detailed analysis in Appendix A.4). Conversely, FELLRec consistently excels, aided by its dynamic balance strategy. 3) FELLRec outperforms all traditional baselines in both centralized and federated settings due to the contextual comprehension and abundant pretrained knowledge of LLMs, along with the dynamic balance strategy.



(a) Reconsturction Similarity Ratio under linear attack (b) Reconsturction Similarity Ratio under MLP attack

Figure 3: (a) and (b) shows the similarity between input embeddings and predicted input embeddings according to extracted embeddings of different layers from BIGRec under linear probe attack and MLP probe attack.

4.3 Attack Model Analysis

To mitigate client-side resource consumption, FELLRec employs the flexible storage strategy in

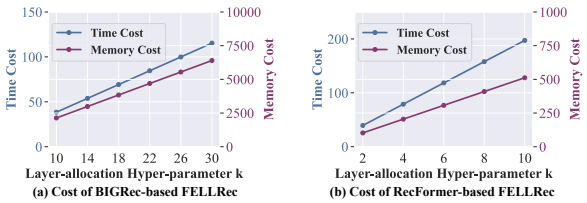


Figure 4: (a) and (b) show the time (s) and memory (MiB) cost for different values of k under the BIGRec-based and RecFormer-based FELLRec, respectively.

Table 3: Cost of FELLRec and FedAvg. k and N represent layer-allocation hyper-parameter and LLM layer number, where $k+1 < N$. b and c represent the communication cost of uploading one layer of LLM parameters and data embeddings to the server, respectively.

	Storage Cost	Inference Cost	Communication Cost
FELLRec	$\mathcal{O}(k+1)$	$\mathcal{O}(k+1)$	$\mathcal{O}((k+1)\cdot b + 2\cdot c)$
FedAvg	$\mathcal{O}(N)$	$\mathcal{O}(N)$	$\mathcal{O}(N\cdot b)$

Section 3.2. However, putting some layers on the server side may also bring the risk of attacking to leak the privacy. In this section, we conduct an attack simulation experiment to assess the possibility of attacks via intermediate embeddings processed on the server side.

We use BIGRec as a case study, extracting intermediate output embeddings from all layers and applying two typical types of white-box attack methods: the linear probe attack and the Multilayer Perceptron (MLP) probe attack (Kim et al., 2024). These methods attempt to reconstruct the input embedding from the layer embeddings separately (Xu et al., 2024a). The detail of the selected attack models is shown in Appendix A.3. We report the cosine similarity ratio between the reconstructed embeddings and the ground truth input embeddings, as illustrated in Figure 3. We find that: 1) the likelihood of reconstructing user historical interactions from intermediate embeddings decreases with ascending layer number generally. 2) The possibility of reconstruction from the last layer increases since LLM training aims to align the final output with the target interacted item, which may have higher similarity with the input embeddings. Thus, the choice of parameter k should be guided by this attack simulation (in this case, $k \geq 21$).

4.4 Efficiency Analysis

We analyze efficiency of FELLRec both experimentally and theoretically. First, we calculate the time and memory cost for different layer-allocation hyper-parameter k in Figure 4. This reveals a trade-off: storing more layers server-side raises the risk of attack but reduces client resource cost. Thus, clients can dynamically adjust layer allocation based on capacity.

We also evaluate FELLRec against FedAvg across various metrics, including storage cost, communication cost, and local client inference cost. The findings in Table 3 demonstrate that our method outperforms FedAvg in storage and inference cost. For communication cost, our method outperforms FedAvg under the conditions: $(k+1)\cdot b + 2\cdot c < n\cdot b$, which simplifies to $c < (n-k-1)b/2$. Intuitively, the lower the value of k, the greater the likelihood of achieving superior communication efficiency compared to FedAvg. Similarly, a lower value of c, indicating a smaller device scope, further enhances the superiority of our method. This indicates that our method is particularly effective in clients with limited scope, making it ideally suited for user devices. Moreover, we can further reduce communication costs for transferring model parameters and data by implementing asynchronous updates (Xu et al., 2023b).

We also conduct a practical analysis of the communication cost associated with uploading a portion of the LLM's parameters to the server. During each communication round, only the LoRA parameters from the client side are uploaded to the server, which significantly reduces the overall parameter size. Since the size of the LoRA parameters for LLaMA-7B is approximately 16 MB (Hu et al., 2021), the communication cost remains manageable. To provide clarity, we quantified the communication cost of uploading LoRA parameters. For LLaMA-7B, the full LoRA parameter size is approximately 16 MB per client. Assuming a 100 Mbps network (a common configuration), the upload time is approximately 1.28 seconds (McMahan et al., 2017). However, with our Flexible Storage Strategy, only the saved layer parameters from clients need to be uploaded, further reducing the communication cost. Moreover, considering that a typical training iteration for LLaMA-7B takes approximately 110 seconds per batch in our experiment, this communication time is negligible in comparison to the overall training time.

4.5 Client Performance Analysis

To assess whether FELLRec mitigates the performance imbalance among various clients, we conduct client evaluation experiment, as detailed in Table 4. Similar results are seen with the Book dataset, but figures are omitted for brevity. The imbalance degree is calculated as follows:

Table 4: Client evaluation results of the centralized method, FedAvg and FELLRec. Bold represents the lowest degree of imbalance among the methods evaluated, using the same backend model.

Recall@10	Games						MicroLens					
	Client 1	Client 2	Client 3	Client 4	Client 5	Imbalance	Client 1	Client 2	Client 3	Client 4	Client 5	Imbalance
BIGRec	0.0227	0.0338	0.0144	0.0163	0.0153	1.35	0.0148	0.0275	0.0059	0.0050	0.0031	4.50
+FedAvg	0.0157	0.0208	0.0235	0.0085	0.0127	1.76	0.0010	0.0047	0.0017	0.0001	0.0004	46.00
+FELLRec	0.0171	0.0211	0.0163	0.0136	0.0152	**0.55**	0.0170	0.0120	0.0066	0.0042	0.0062	**3.04**

Imbalance Degree $= (m_{\text{best}} - m_{\text{worst}})/m_{\text{worst}}$, where m_{best} is the Recall@10 of the best client, and m_{worst} is the Recall@10 of the worst client. The results indicate that: 1) FELLRec effectively mitigates the performance imbalance issue among clients compared to FedAvg, primarily due to the dynamic balance strategy, which customizes dynamic parameter aggregation and learning speed for different clients. 2) The imbalance degree in the MicroLens dataset is more pronounced under FedAvg, which is potentially caused by the data distribution among clients being much more diverse than others. Such diversity may lead to conflict optimization objectives among clients, thus exacerbating the imbalance.

• **More In-depth Experiments.** Ablation study, client heterogeneity analysis, client number study, and hyper-parameter analysis are in Appendix A.4.

5 Related Work

• **LLM-based Recommendation.** Recent advances in LLMs for recommendation systems have gained attention for their contextual understanding and pre-trained knowledge (Lin et al., 2024). Early efforts, such as P5 (Geng et al., 2022), TALLRec (Bao et al., 2023b), focused on fine-tuning LLMs with prompts and recommendation data. Later works, like BIGRec (Bao et al., 2023a) and TIGER (Rajput et al., 2023), refine LLMs by grounding outputs in real item spaces and enhancing generative processes with semantic information. This shift moves from simply integrating recommendation data to fully leveraging LLMs for improved performance. As performance improves, the focus expands to include trustworthiness, such as fairness and explainability (Zhang et al., 2023c; Wang et al., 2023a). Studies like (Xu et al., 2024a; Dai et al., 2024; Xu et al., 2024b) highlight user unfairness in LLM-based recommendation, and LLMHG (Chu et al., 2024) introduces an explainable framework combining LLM reasoning with hypergraph neural networks.

• **Federated Recommendation.** Fed4Rec enhances data privacy in recommendation systems using federated learning (Yin et al., 2024; Zhang et al., 2024), operating under two main frameworks: 1) **Peer-Peer Framework** (Yang et al., 2022; Long et al., 2023a,b): Clients directly broadcast intermediate parameters to other clients, who aggregate them into their models. For example, SemiDFEGL(Qu et al., 2023) improves scalability via device-to-device collaboration, while DGRec (Zheng et al., 2023) uses a decentralized graph neural network. However, this framework has high communication costs due to large LLM parameters. 2) **Client-Server Framework** (Wang et al., 2022; Liu et al., 2023; Imran et al., 2023; Zhang et al., 2023a): Clients send local parameters to a central server for aggregation and redistribution. Examples include FedPerGNN(Wu et al., 2022), which incorporates high-order information while preserving privacy, and LightFR (Zhang et al., 2023b), a lightweight federated matrix factorization framework with efficient inference.

6 Conclusion

In this work, we proposed a federated framework for LLM-based recommendation (FELLRec). Firstly, we identified two key challenges in directly applying Fed4Rec in the LLM-based recommendation: exacerbated client performance imbalance and high client resource costs. Subsequently, to address these, FELLRec introduces: 1) dynamic balance strategy, which designs dynamic parameter aggregation and learning speed for different clients during training, aims to ensure relatively equitable performance across clients. 2) Flexible storage strategy, which selectively retains certain sensitive LLM layers on the client side, while offloading other layers to the server, aims to save resources. Overall, FELLRec offers an equitable and resource-efficient approach to safeguard data privacy in LLM-based recommendations.

Acknowledgments

This research/project is supported by the National Research Foundation Singapore and DSO National Laboratories under the AI Singapore Programme (AISG Award No: AISG2-RP-2020-018)

Limitations

First, the largest model we use in this work is LLaMA-7B, exploring the potential of using even larger LLMs could provide further insights into the effectiveness of our method. Second, we primarily utilize two common types of white-box attack methods for model analysis. Additionally, we only use BIGRec as a case study for these attacks. However, different LLMs may demonstrate varying levels of resilience to different attack methods. In the future, it would be beneficial to apply a broader range of attack methods across various LLM architectures to further validate the effectiveness of our approach. Third, while the dynamic balance strategy we designed for FELLRec has proven effective, it would be promising to explore more fine-grained aggregation strategies (*e.g.,* layer-based aggregation) in the future.

References

Jingmin An, Guanyu Li, and Wei Jiang. 2024. Nrdl: Decentralized user preference learning for privacy-preserving next poi recommendation. Expert Systems with Applications, 239:122421.

Keqin Bao, Jizhi Zhang, Wenjie Wang, Yang Zhang, Zhengyi Yang, Yancheng Luo, Fuli Feng, Xiangnaan He, and Qi Tian. 2023a. A bi-step grounding paradigm for large language models in recommendation systems. arXiv:2308.08434.

Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. 2023b. Tallrec: An effective and efficient tuning framework to align large language model with recommendation. In RecSys. ACM.

Robin Burke, Nasim Sonboli, and Aldo Ordonez-Gauger. 2018. Balanced neighborhoods for multi-sided fairness in recommendation. In Conference on fairness, accountability and transparency, pages 202–214. PMLR.

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

Chaochao Chen, Xiaohua Feng, Jun Zhou, Jianwei Yin, and Xiaolin Zheng. 2023. Federated large language model: A position paper. arXiv:2307.08925.

Hong Chen, Yudong Chen, Xin Wang, Ruobing Xie, Rui Wang, Feng Xia, and Wenwu Zhu. 2021. Curriculum disentangled recommendation with noisy multi-feedback. Advances in Neural Information Processing Systems, 34:26924–26936.

Shuangyi Chen, Yue Ju, Hardik Dalal, Zhongwen Zhu, and Ashish Khisti. 2024. Robust federated finetuning of foundation models via alternating minimization of lora. arXiv:2409.02346.

Zhixuan Chu, Yan Wang, Qing Cui, Longfei Li, Wenqing Chen, Sheng Li, Zhan Qin, and Kui Ren. 2024. Llm-guided multi-view hypergraph learning for human-centric explainable recommendation. arXiv preprint arXiv:2401.08217.

Sunhao Dai, Chen Xu, Shicheng Xu, Liang Pang, Zhenhua Dong, and Jun Xu. 2024. Bias and unfairness in information retrieval systems: New challenges in the llm era. In KDD, page 6437–6447. Association for Computing Machinery.

Tao Fan, Yan Kang, Guoqiang Ma, Weijing Chen, Wenbin Wei, Lixin Fan, and Qiang Yang. 2023. Fate-llm: A industrial grade federated learning framework for large language models. arXiv:2310.10049.

Zeno Gantner, Lucas Drumond, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2011. Personalized ranking for non-uniformly sampled items. In KDDCUP, page 231–247. JMLR.

Yunfan Gao, Tao Sheng, Youlin Xiang, Yun Xiong, Haofen Wang, and Jiawei Zhang. 2023. Chat-rec: Towards interactive and explainable llms-augmented recommender system. arXiv:2303.14524.

Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. 2022. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). In RecSys, pages 299–315.

Wes Gurnee and Max Tegmark. 2023. Language models represent space and time. arXiv:2310.02207.

Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In SIGIR, pages 639–648.

Chris Jay Hoofnagle, Bart Van Der Sloot, and Frederik Zuiderveen Borgesius. 2019. The european union general data protection regulation: what it is and what it means. Information & Communications Technology Law, 28(1):65–98.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. arXiv:2106.09685.

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. TOIS, 41(3):1–30.

Apple Inc. 2024. Introducing apple's on-device and server foundation models.

Siwon Kim, Sangdoo Yun, Hwaran Lee, Martin Gubri, Sungroh Yoon, and Seong Joon Oh. 2024. Propile: Probing privacy leakage in large language models. Advances in Neural Information Processing Systems, 36.

Jiacheng Li, Ming Wang, Jin Li, Jinmiao Fu, Xin Shen, Jingbo Shang, and Julian McAuley. 2023. Text is all you need: Learning language representations for sequential recommendation. arXiv:2305.13731.

Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. 2021. Ditto: Fair and robust federated learning through personalization. In International conference on machine learning, pages 6357–6368. PMLR.

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

Xinyu Lin, Wenjie Wang, Yongqi Li, Shuo Yang, Fuli Feng, Yinwei Wei, and Tat-Seng Chua. 2024. Data-efficient fine-tuning for llm-based recommendation. arXiv:2401.17197.

Ruixuan Liu, Yang Cao, Yanlin Wang, Lingjuan Lyu, Yun Chen, and Hong Chen. 2023. Privaterec: Differentially private model training and online serving for federated news recommendation. In SIGKDD, pages 4539–4548.

Jing Long, Tong Chen, Quoc Viet Hung Nguyen, Guandong Xu, Kai Zheng, and Hongzhi Yin. 2023a. Model-agnostic decentralized collaborative learning for on-device poi recommendation. In SIGIR, pages 423–432.

Jing Long, Tong Chen, Quoc Viet Hung Nguyen, and Hongzhi Yin. 2023b. Decentralized collaborative learning framework for next poi recommendation. TOIS, 41(3):1–25.

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, pages 1273–1282. PMLR.

Khalil Muhammad, Qinqin Wang, Diarmuid O'Reilly-Morgan, Elias Tragos, Barry Smyth, Neil Hurley, James Geraci, and Aonghus Lawlor. 2020. Fedfast: Going beyond average for faster training of federated recommender systems. In KDD, pages 1234–1242.

Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. 2017. Practical black-box attacks against machine learning. In Proceedings of the 2017 ACM on Asia conference on computer and communications security, pages 506–519.

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. arXiv:2302.10900.

Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan H Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Q Tran, Jonah Samost, et al. 2023. Recommender systems with generative retrieval. In NeurIPS. Curran Associates, Inc.

Zehua Sun, Yonghui Xu, Yong Liu, Wei He, Lanju Kong, Fangzhao Wu, Yali Jiang, and Lizhen Cui. 2022. A survey on federated recommendation systems. arXiv:2301.00767.

Lei Wang, Songheng Zhang, Yun Wang, Ee-Peng Lim, and Yong Wang. 2023a. Llm4vis: Explainable visualization recommendation using chatgpt. arXiv:2310.07652.

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, 31(5):877–896.

Song Wang, Xingbo Fu, Kaize Ding, Chen Chen, Huiyuan Chen, and Jundong Li. 2023b. Federated few-shot learning. In KDD, pages 2374–2385.

Chuhan Wu, Fangzhao Wu, Lingjuan Lyu, Tao Qi, Yongfeng Huang, and Xing Xie. 2022. A federated graph neural network framework for privacy-preserving personalization. Nature Communications, 13(1):3091.

Chen Xu, Sirui Chen, Jun Xu, Weiran Shen, Xiao Zhang, Gang Wang, and Zhenhua Dong. 2023a. P-mmf: Provider max-min fairness re-ranking in recommender system. In Proceedings of the ACM Web Conference 2023, WWW '23, page 3701–3711, New York, NY, USA. Association for Computing Machinery.

Chen Xu, Wenjie Wang, Yuxin Li, Liang Pang, Jun Xu, and Tat-Seng Chua. 2024a. A study of implicit ranking unfairness in large language models. In EMNLP Findings, pages 7957–7970. Association for Computational Linguistics.

Chen Xu, Xiaopeng Ye, Wenjie Wang, Liang Pang, Jun Xu, and Ji rong Wen. 2024b. A taxation perspective for fair re-ranking. In SIGIR. Association for Computing Machinery.

Chenhao Xu, Youyang Qu, Yong Xiang, and Longxiang Gao. 2023b. Asynchronous federated learning on heterogeneous devices: A survey. Computer Science Review, 50:100595.

He Yang, Wei Xi, Zizhao Wang, Yuhao Shen, Xinyuan Ji, Cerui Sun, and Jizhong Zhao. 2023. Fedrich: Towards efficient federated learning for heterogeneous clients using heuristic scheduling. Information Sciences, 645:119360.

Xu Yang, Yuchuan Luo, Shaojing Fu, Ming Xu, and Yingwen Chen. 2022. Dpmf: Decentralized probabilistic matrix factorization for privacy-preserving recommendation. Applied Sciences, 12(21):11118.

Hongzhi Yin, Liang Qu, Tong Chen, Wei Yuan, Ruiqi Zheng, Jing Long, Xin Xia, Yuhui Shi, and Chengqi Zhang. 2024. On-device recommender systems: A comprehensive survey. arXiv:2401.11441.

Chunxu Zhang, Guodong Long, Tianyi Zhou, Peng Yan, Zijian Zhang, Chengqi Zhang, and Bo Yang. 2023a. Dual personalization on federated recommendation. arXiv:2301.08143.

Honglei Zhang, He Liu, Haoxuan Li, and Yidong Li. 2024. Transfr: Transferable federated recommendation with pre-trained language models. arXiv:2402.01124.

Honglei Zhang, Fangyuan Luo, Jun Wu, Xiangnan He, and Yidong Li. 2023b. Lightfr: Lightweight federated recommendation with privacy-preserving matrix factorization. TOIS, 41(4):1–28.

Jizhi Zhang, Keqin Bao, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. 2023c. Is chatgpt fair for recommendation? evaluating fairness in large language model recommendation. In RecSys. ACM.

Xiaolin Zheng, Zhongyu Wang, Chaochao Chen, Jiashu Qian, and Yao Yang. 2023. Decentralized graph neural network for privacy-preserving recommendation. In CIKM, pages 3494–3504.

A Appendix

A.1 Baselines

1) **BIGRec** (Bao et al., 2023a) using LLaMA-7B as the LLM backbone, utilizing the item title to present the user sequence. 2) **RecFormer** (Li et al., 2023) using LongFormer as the LLM backbone, utilizing both item titles and descriptions to represent user sequences. 3) **FedAvg** (McMahan et al., 2017) aggregates client model parameters without uploading their data. 4) **FedProx** (Li et al., 2020) extends FedAvg by adding a proximity term to the local optimization, allowing for more robust handling of heterogeneous data across clients. 5) **Ditto** is a personalized federated learning framework that simultaneously ensures fairness and robustness in statistically heterogeneous networks via a scalable solver. 6) **RoLoRA** employs an alternating minimization approach for LoRA to enhance robustness against reduced fine-tuning parameters and heightened data heterogeneity. 7) **MF** (Gantner et al., 2011) is a classical matrix factorization (MF) approach. 8) **LightGCN** (He et al., 2020) leverages high-order neighbor information to enhance the user and item representations. 9) **FedMF** (Chai et al., 2020) is a privacy-enhanced MF approach based on secure homomorphic encryption. 10) **LightFR** (Zhang et al., 2023b) is a lightweight federated recommendation framework with privacy-preserving MF. 11) **FedPerGNN** (Wu et al., 2022) designs a privacy-preserving graph expansion protocol to incorporate high-order information under privacy protection in GNN-based recommendation.

A.2 Implementation

For all the baselines, we follow the original settings in their paper for implementation. Besides, we adopt the parameter-efficient fine-tuning technique LoRA to fine-tune BIGRec in 1024-shot and fully fine-tune RecFormer. For the client partition, we set the client number equal to 5, and cluster users based on pre-trained MF user embeddings, leveraging the premise that users with analogous characteristics and preferences are more likely to congregate in similar areas or platforms. For FedAvg, FedProx and FELLRec, we set the same local round number to ensure a fair comparison. The best hyper-parameters are selected with the searching scopes as follows: speed-related warm-up factor and time-related warm-up factor are tuned in $\{0.1, 0.3, 0.5, 0.7, 0.9, 1.1, 1.3\}$ and $\{1, 3, 5, 10, 15, 20\}$. The experiments were conducted under four NVIDIA V100.

Table 5: Statistics of three datasets.

Dataset	#User	#Item	#Interaction	Density
MicroLens	45,886	12,413	332,730	5e-04
Games	50,532	16,857	452,894	5e-04
Book	64,989	56,394	4,963,757	1.3e-03

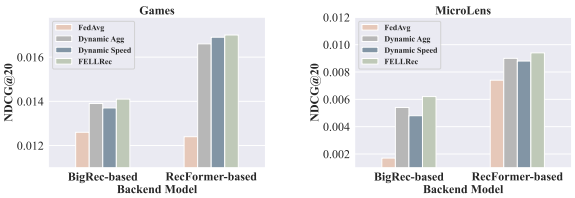


Figure 5: Contributions of dynamic parameter aggregation and learning speed to FELLRec.

A.3 Attack Model Selection

For the attack model, there are two kinds of attack models: white-box (Gurnee and Tegmark, 2023) and black-box (Papernot et al., 2017). White-box means the attacker has complete knowledge of the target model, including the model's architecture, parameters, and training data. Black-box means attackers have very limited knowledge of the target model. They may only be able to speculate about the model's partial behavior through its inputs and outputs. To validate the robustness of our approach, we conducted experiments involving two typical types of white-box attacks: the linear probe attack and the Multilayer Perceptron (MLP) probe attack (Kim et al., 2024). These methods have better attack capabilities than black-box methods due to their access to more prior knowledge.

The linear probe attack involves training a linear regression model on the intermediate output embeddings of LLM. This model attempts to recover the original data from these intermediate embeddings. We then compare the similarity ratio between the reconstructed embeddings and the ground truth input embeddings, where a lower similarity ratio indicates greater difficulty for the attack model to extract useful information. The MLP probe attack is similar to the linear probe attack but employs a Multilayer Perceptron (MLP) instead of a simple linear regression model to probe the intermediate representations.

A.4 In-depth Experimental analysis

A.4.1 Ablation Study

In this section, we evaluate the unique contributions of dynamic parameter aggregation and dynamic learning speed in comparison with FedAvg,

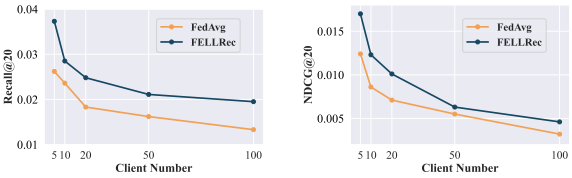


Figure 6: Influence of Client Number to RecFormer-based FELLRec and FedAvg in Games.

Table 6: Performance (R@10) of FedAvg and FELLRec under different heterogeneity degrees in Games. As concentration parameter c increases, the heterogeneity degree decreases.

c	**0.1**	**0.3**	**0.5**	**0.7**	**1**
FedAvg	0.0134	0.0139	0.0141	0.0157	0.0162
FELLRec	0.0187	0.0211	0.0212	0.0216	0.0228

presenting the results in Figure 5 for the Games and Microlens datasets (excluding the Book dataset due to analogous trends). The analysis reveals that: 1) FELLRec with dynamic parameter aggregation consistently surpasses FedAvg. This highlights the benefits of an attention-based parameter aggregation method that tailors aggregation to the specific data distribution within FELLRec. 2) Similarly, FELLRec with dynamic learning speed invariably outperforms FedAvg, emphasizing the advantages of customizing the learning speed of each client based on their learning status. 3) The effectiveness of the two parts is consistent across different datasets and backend models, further demonstrating their robustness and generalizability.

A.4.2 Influence of Client Heterogeneity Degree

To further demonstrate the robustness of our framework under different data distributions, we conduct additional experiments to analyze the influence of the heterogeneous data distribution degree across clients on the performance of federated learning baselines and our proposed method. Specifically, we follow the prevailing strategy (Wang et al., 2023b) and distribute samples to all clients based on the Dirichlet distribution. We report the performance under different heterogeneity degrees in Table 6, with c as the concentration parameter determining the heterogeneity degree across clients. Intuitively, as the concentration parameter c increases, the heterogeneity degree decreases. The results indicate that: 1) FELLRec consistently outperforms FedAvg when using the same backend model (BIGRec), demonstrating the superior capability of the dynamic balance strategy under different heterogeneity degree settings. 2) As the heterogeneity degree increases, the performance of FedAvg decreases, corroborating the conclusion from Section 4.2 that the heterogeneity degree significantly impacts FedAvg's performance, causing fluctuations under different data distributions across clients.

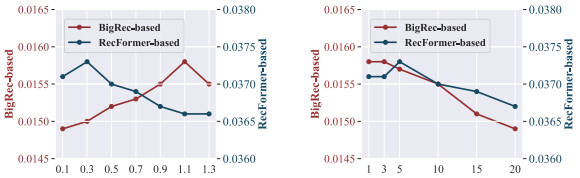


Figure 7: Hyper-parameters analysis of Games.

A.4.3 Influence of Client Number

To demonstrate the scalability of our approach with an increased number of clients, we expanded the client count from 5 to 100 and reported the comparative results of FELLRec and FedAvg in Figure 6. The findings indicate that: 1) With the escalation in client numbers, there is a noticeable decline in the performance of both FELLRec and FedAvg, likely due to the amplified diversity acorss client data distribution, which in turn aggravates the imbalance and adversely affects overall performance. 2) Nevertheless, FELLRec consistently outperforms FedAvg in every client count scenario. This enhanced performance is attributed to the dynamic aggregation strategy employed by FELLRec, effectively countering the imbalances stemming from the varied data distributions among clients.

A.4.4 Hyper-parameter Analysis

We select sensitive hyper-parameters, adjusting them within the ranges delineated in Section 4.1. The experiment outcomes are visually represented in Figure 7. From our observations: The settings of the speed-related warm-up factor α and the time-related warm-up factor β significantly affect the warm-up speed. Generally, enhancing the values of α and β leads to improved performance, facilitating the integration of parameters from other clients to aid the learning process of the current client once it has adequately learned from its data. Nevertheless, overly aggressive acceleration in warm-up may prematurely incorporate parameters from other clients before the current client is prepared, potentially disrupting the learning trajectory and adversely affecting performance.