Knowledge-Based Systems 255 (2022) 109483

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Federated low-rank tensor projections for sequential recommendation

Li Li [a], Fan Lin [a], Jianbing Xiahou [a,b,*], Yuanguo Lin [a], Pengcheng Wu [c], Yong Liu [c]

[a] School of Informatics, Xiamen University, Xiamen, China
[b] Quanzhou Normal University, Quanzhou, China
[c] Joint NTU-UBC Research Centre of Excellence in Active Living for the Elderly (LILY), Nanyang Technological University, Singapore

ARTICLE INFO

Article history:
Received 22 July 2021
Received in revised form 22 June 2022
Accepted 15 July 2022
Available online 22 July 2022

Keywords:
Federated learning
Federated recommendation
Sequential recommendation

ABSTRACT

Recommender systems have achieved great success in many fields, benefiting from the accessibility of massive amounts of user behavior data. However, the increasing concern about privacy issues brings new research challenges to traditional data-centralized recommender systems. As a privacy-preserving machine learning technique, federated learning provides potential solutions to these research challenges, by jointly training perceptive global recommendation models based on the decentralized user behavior data. Although existing federated recommendation methods have achieved promising results, they still suffer the following limitations. Firstly, they mainly focus on the horizontal federated recommendation scenarios where the participants are individual users, without considering the scenarios where participants are organizations. Secondly, they tend to model static user preferences and cannot learn the preference evolution in users' interaction sequences. To address these limitations, we propose a horizontal Federated recommendation framework for Sequential Recommendation (FedSeqRec), where the participants are organizations. As the data in different organizations are usually not independent and identically distributed (Non-IID), federated training tends to yield suboptimal results. To alleviate this problem, we propose a federated re-average algorithm, which takes the consistency of the client's data distribution with the overall data distribution into consideration. Moreover, we also leverage a low-rank tensor projection to model user long-term preferences. Experimental results on two real-world datasets demonstrate that FedSeqRec outperforms state-of-the-art federated recommendation methods. The implementation code of FedSeqRec is available at https://github.com/MuziLee-x/FedSeqRec.

1. Introduction

Recommender systems play a crucial role in our daily lives, such as course recommendations on online learning platforms [1–3], point-of-interest recommendations in location-based services [4,5], and micro-video recommendations on E-commerce platforms [6]. The recent successes of recommender systems originate from sophisticated neural models [7] and the large amount of user behavior data collected in different application scenarios. However, under the increasingly stringent data protection regulations, *e.g.*, General Data Protection Regulation (GDPR) [8], collecting all user activity data to train recommendation models is becoming infeasible. Nevertheless, Federated Learning (FL) [9–11] as a privacy-preserving machine learning framework can alleviate this issue. Thus, by combining the advantages of recommender systems and federated learning, the federated recommender systems [12–16] have emerged.

In general, current federated recommendation methods usually adopt the federated learning framework comprising a cloud server and massive mobile clients, each client representing an individual user device. Different clients share the same feature space but different samples. For example, [12] presents the first federated implementation of a collaborative filtering-based recommendation model, where each user's raw data is locally stored and employed to calculate the local gradients. In [16], the authors propose a distributed privacy-preserving news recommendation model, in which the clients deploy the global model locally, compute local gradients based on local data, and send the local gradients to the central server for further aggregation. They adopt horizontal federated learning frameworks [9], and the participants in these methods are individual users.

Differing from previous methods, this paper focuses on a new federated recommendation scenario, where the participants in the horizontal FL framework are organizations. This setting exists in many real application scenarios. For instance, two regional service companies with similar businesses are located in different

* Corresponding author at: School of Informatics, Xiamen University, Xiamen, China.
E-mail addresses: lylee@stu.xmu.edu.cn (L. Li), iamafan@xmu.edu.cn (F. Lin), jbxiahou@xmu.edu.cn (J. Xiahou), xdlyg@stu.xmu.edu.cn (Y. Lin), pengcheng.wu@ntu.edu.sg (P. Wu), stephenliu@ntu.edu.sg (Y. Liu).

https://doi.org/10.1016/j.knosys.2022.109483
0950-7051/

states. The customer data in each company is not adequate to train a complex model. Thus, there is a need to train an intelligent model by exploiting all the data of these two companies. Due to the competition and the local consumer privacy regulations, such as GDPR [8], they need to store their customer data locally and not exchange sensitive data with each other. In the new federated recommendation scenario, the data in different organizations tend to be not independent and identically distributed (Non-IID). The well-known Federated Averaging (FedAvg) strategy [17] alleviates the convergence problem caused by the Non-IID data to some extent. However, it ignores the deviation of the local organization data distribution from the overall distribution, thus making the global model training converge to sub-optimal results.

Moreover, existing federated recommendation methods usually utilize user interaction data to learn static preferences. Hence, they may be unsuitable for scenarios where user preferences evolve. For example, on Massive Open Online Courses (MOOCs) platforms, a user's learning process is dynamically shifting due to his/her learning preferences evolution and the prerequisite relation between knowledge concepts contained in the courses [18]. Similarly, a user's preferences on movies may also be dynamic, potentially changing with his/her preferences or external influences such as a friend's suggestion [19]. Thus, studying the federated sequential recommendation problem is necessary. In the latter scenario, the user's behavior shows to some degree non-exchangeability in time [20]. In mathematics, using non-commutative algebras to describe sequences is a classic problem [21] that affords an appealing solution to time series classification tasks [22]. This indicates that the algebra method may be suitable for federated sequential recommendation tasks. However, using algebra-based methods to model user dynamic preference has been rarely explored.

To this end, we propose federated low-rank tensor projections for sequential recommendation, named Federated Sequential Recommendation (FedSeqRec), where the participants are organizations. To alleviate the Non-IID problem, we present a federated re-averaging algorithm. Concretely, given that FedAvg considers the organizations' data participating in each round, the federated re-average algorithm additionally considers the deviation between the organizational and the overall data distributions. Besides, we propose adopting low-rank tensor projection [22] to model the evolution of user preferences.

The main contributions of this paper are summarized as follows:

- We propose a novel federated sequential recommendation method, where several organizations can effectively collaborate to train an intelligent sequential recommendation model.
- We generalize FedAvg based on the similarity between the organization and the overall data distributions.
- We develop an algebra-based sequential recommendation model that utilizes low-rank tensor projections to model the dynamic preferences of the users.
- We perform comprehensive experiments on real-world datasets to illustrate the effectiveness of FedSeqRec and challenge it against traditional centralized sequential recommendation methods and federated recommendation methods.

The remaining content is organized as follows. Section 2 reviews the most relevant existing work. Section 3 offers the details of the proposed FedSeqRec. Next, in Section 4, we display the experimental results. Finally, we conclude this work and discuss the future directions in Section 5.

2. Related work

We summarize the most relevant existing work in the following research fields, *i.e.*, federated learning, federated recommendation, and sequential recommendation.

2.1. Federated learning

Federated Learning (FL) [9] is a privacy-aware machine learning technique that jointly trains perceptive global models based on decentralized user data [17]. Compared with traditional centralized machine learning techniques, FL allows the raw data locally stored instead of pulling all user data together. When the global model starts training, each client device copies the initialized global model as its local model and trains it regionally. Then, the parameters or gradients of the local model are sent to a central server. After receiving the local models, the central server aggregates them into a unified one. Again, all clients copy the global model from the server and perform local updates. This process will continue until the global model converges. Although federated learning shares some similar spirits with distributed learning, they are inherently different. In distributed learning, the data is centrally stored in the server and can be shuffled and balanced across the workers [23]. Namely, the data in different clients are IID. On the contrary, in federated learning, each client's raw data is stored locally and not exchanged or transferred [23]. In other words, the sensitive data of different clients tend to be Non-IID. According to the datasets' distribution characteristics belonging to different data owners, existing federated learning methods can be classified into three main categories: horizontal federated learning, vertical federated learning, and federated transfer learning [9]. More details about federated learning can be found in recent survey papers [10,11].

2.2. Federated recommendation

Recently, federated learning has been applied successfully in the development of recommender systems [12–16,24–26]. Following the taxonomy of traditional recommendation methods, existing federated recommendation models can be classified into Collaborative Filtering (CF)-based methods [12–15] and content-based methods [16]. For example, in CF-based methods, FCF [12] adapts traditional CF [27] to assist federated learning to learn the embedding of users on clients and items on the server iteratively. FMF [13] further protects item gradients with homomorphic encryption based on FCF. Moreover, the FedRec++ model [14] employs PMF [28] as the global model to train a lossless federated recommendation model by allocating ordinary clients and denoising clients. In content-based methods, FedNewsRec [16] is the first content-based federated recommendation model designed for news recommendation. To summarize, existing federated recommendation methods are horizontal frameworks where the participants are mobile clients. They mainly exploit static user behaviors to learn user embeddings and cannot effectively capture the evolution of user preferences in federated sequential recommendation scenarios.

2.3. Sequential recommendation

The sequential recommender systems (SRSs) exploit the interaction sequences between users and items to reveal the users' next behavior [29–31].

In practice, an effective way to model user preference is adopting a Recurrent Neural Network (RNN), *e.g.*, Gated Recurrent Unit (GRU) [32], and Long Short-Term Memory (LSTM) network [33], to represent a user interaction sequence as a vector [34–36].

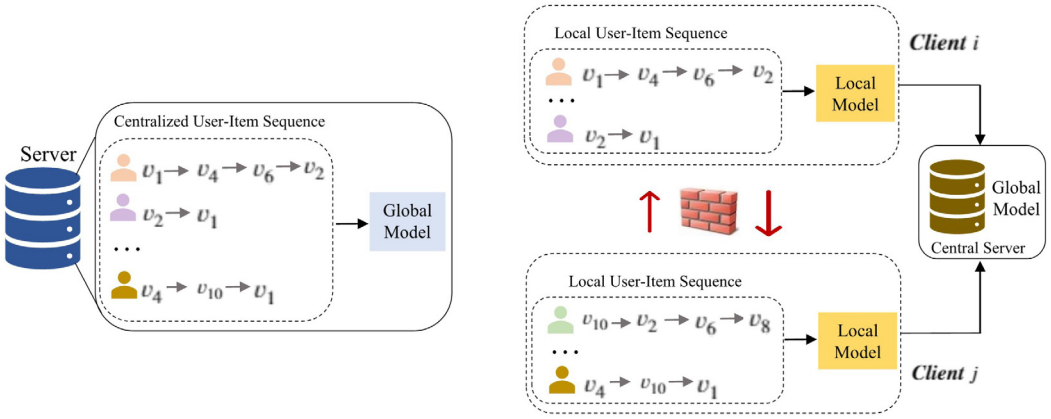


Fig. 1. Comparisons between traditional centralized sequential recommendation methods and decentralized way. **Left**: Traditional centralized sequential recommendation framework. **Right**: Decentralized federated sequential recommendation framework.

Representative methods are GRU4Rec [32], Dynamic GRU [36], and the user-based GRU [37]. Besides RNN-based methods, Graph Neural Networks (GNNs) have also demonstrated superior performances in capturing the dependency in users' behaviors [38–40]. For instance, the SR-GNN model [38] analyzes complex item transitions within a sequence by exploiting gated graph neural networks. In addition, the attention mechanism can boost the sequential recommendation performances [41,42,42–46]. For instance, the SASRec model [43] employs a two-layer Transformer [47] decoder to model user sequences, and BERT4Rec [44] utilizes a bidirectional method to model the users' sequential behaviors. GC-SAN [39] and FGNN [40] enhance SR-GNN by adding a self-attention mechanism and weighted graph attention network, respectively. In summary, these methods adopt centralized training methods, which may cause the risk of data privacy leakage.

3. The proposed recommendation model

In this section, we first introduce the problem studied in this work. Then, we present the details of the proposed FedSeqRec model. Next, we discuss the privacy-preserving mechanism of FedSeqRec.

3.1. Problem formulation

This work studies the sequential recommendation problem under the federated setting. Fig. 1 illustrates the key differences between a traditional centralized sequential recommendation and a federated sequential recommendation scheme. As depicted in Fig. 1 (left part), conventional sequential recommendation models usually require pooling the raw user–item interactions to train the global model. In this scenario, it is easy to induce data privacy leakage in transmitting raw user–item interaction data. From the right part of Fig. 1, we note that a potential solution to this problem is preserving the raw data locally in each organization and employing a decentralized strategy to train a global sequential recommendation model.

Suppose the federated recommender system involves M users and N items, and we use $\mathcal{U}$ and $\mathcal{V}$ to denote the user set and item set, respectively. The users' interaction sequence data are scattered on C clients $\mathcal{C} = \{c_k\}_{k=1}^{k=C}$, which are isolated. Let $\mathcal{U}_k$ represent the set of users located on the client c_k. For each user u, we use $\mathcal{S}_u$ to denote her interaction sequence, located at a specific client. In this federated sequential recommendation task, all clients jointly train a sequential recommendation model leveraging the union of their data without pulling them together.

Table 1
Main notations.

Notations	Description
$\mathcal{V}$	The set of items, $\mathcal{V} = \{v_1, v_2, \ldots, v_N\}$
v_i	The ith item of $\mathcal{V}$, $1 \le i \le N$
$\mathcal{U}$	The set of users, $\mathcal{U} = \{u_1, u_2, \ldots, u_M\}$
u	The uth user of $\mathcal{U}$, $1 \le u \le M$
$\mathcal{C}$	The set of clients, $\mathcal{C} = \{c_1, c_2, \ldots, c_C\}$
c_j	The jth client of $\mathcal{C}$, $1 \le j \le C$
$\mathcal{U}_k$	The set of users at the kth client
$\mathcal{S}_u$	Behaviors of user $u \in \mathcal{U}$, $\mathcal{S}_u = [v_1^u, v_2^u, \ldots, v_{n_u}^u]$

To be consistent with the terminology in federated learning, in the following content, we use the term *client* to refer to the word *organization* mentioned above.

The main notations used in this work are summarized in Table 1.

3.2. FedSeqRec framework

The overall framework of the proposed FedSeqRec is illustrated in Fig. 2. Differing from the classic four-step federated learning framework [9], FedSeqRec is a six-step federated learning framework. It comprises a central server and several clients. The central server is responsible for the whole model training process. Each client receives the initialized or updated global model from the central server, trains a local model from its user interaction data, and sends the local parameters to the server. After receiving the local models, the central server aggregates them into a unified model and delivers the updated global model to clients. This process loops several times until the global model converges. Next, we describe the details of each core step.

3.2.1. Update local data distribution

For client c_k, we abstract the interaction sequence length of the user located on c_k as a random variable $\mathbf{X}_k$ that obeys a polynomial distribution, where the number of its parameters is the sum of elements in the set formed by the length of interaction sequence of all users.

Specifically, the client c_k counts the length of the user sequences based on its local data to form a local list of user sequence lengths denoted as $\mathcal{S}^k$, and updates $\mathcal{S}^k$ to the server. After receiving $\mathcal{S}^k$ from clients $\mathcal{C} = \{c_k\}_{k=1}^{k=C}$, the server takes their union and sorts them in ascending order to form a global list of user sequence lengths denoted as $\mathcal{S}$, and sends $\mathcal{S}$ to all clients.

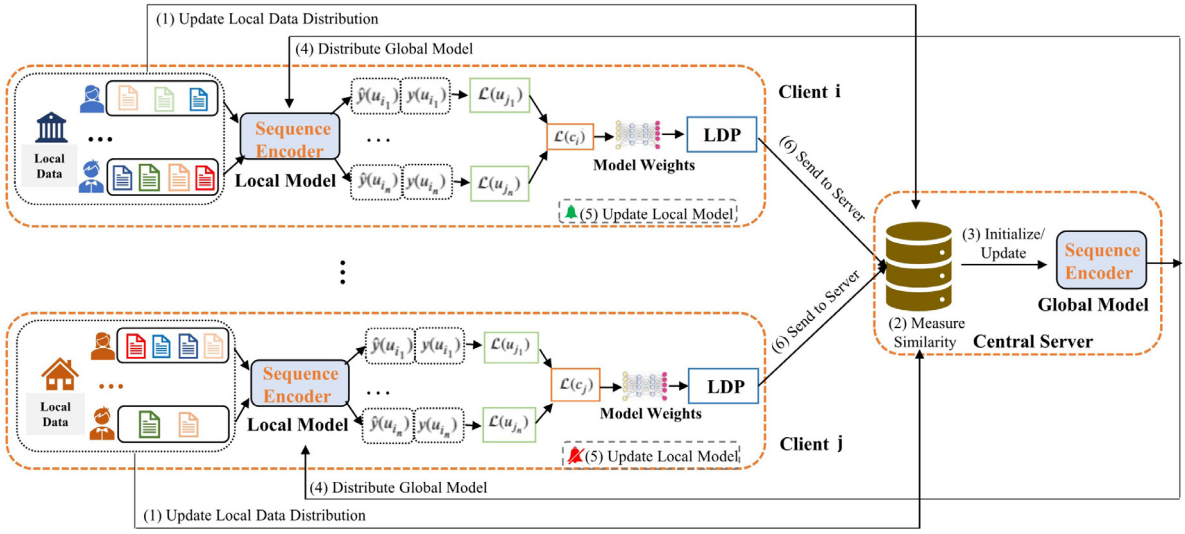


Fig. 2. The overall framework of **FedSeqRec**. The pipeline mainly consists of six steps: (1) clients send their local data distribution to the central server; (2) the central server measures the similarity between the overall data distribution and the client data distribution; (3) the central server initializes/updates the global model; (4) the central server distributes the global model to clients; (5) clients update their local models, and (6) clients send their local updates to the central server.

Namely, the first element $\mathcal{S}_{[1]}$ of $\mathcal{S}$ is the minimum length of the user sequence, and the second element $\mathcal{S}_{[2]}$ of $\mathcal{S}$ is the second smallest length of the user sequence. After receiving $\mathcal{S}$ from the server, the client c_k estimates the parameters of the distribution $\mathbf{X}_k$ based on its local data and provides $\mathbf{X}_k \sim P(\hat{\mathrm{p}}_{\mathrm{k}_1}, \hat{\mathrm{p}}_{\mathrm{k}_2}, \ldots, \hat{\mathrm{p}}_{\mathrm{k}_n})$ where $k_1 = \mathcal{S}_{[1]}, k_2 = \mathcal{S}_{[2]}, \ldots, k_n = \max\{|\mathcal{S}_u|, \forall u \in \mathcal{U}\}$. In other words, for the kth client, $\hat{p}_{k_j}$ is the probability estimation when the random variable $\mathbf{X}_k$ takes k_j. Subsequently, the client c_k updates its estimated parameters, $\hat{\mathrm{p}}_{\mathrm{k}_1}, \hat{\mathrm{p}}_{\mathrm{k}_2}, \ldots, \hat{\mathrm{p}}_{\mathrm{k}_n}$, to the central server.

All clients compute and update their local data distribution to the central server in parallel.

3.2.2. Measure similarity

Let $\mathbf{X}_c$ be a random variable that represents the length of the interaction sequence of the users located at the central server c. By applying the Kullback–Leibler divergence, we obtain the similarity between the kth client c_k and the central server c as follows:

$$Sim(c_k, c) = \sum_{i \in \mathcal{S}} p(\mathbf{X}_k = i) \log \frac{p(\mathbf{X}_k = i)}{q(\mathbf{X}_c = i)}, \tag{1}$$

where $p(\mathbf{X}_k = i)$ and $q(\mathbf{X}_c = i)$ is the estimated probability that the length of the user interaction sequence is i on kth client and the central server, respectively. After normalization, we obtain the similarity adjustment factor γ_k on the kth client:

$$\gamma_k = \frac{\exp\left(-Sim(c_k, c)\right)}{\sum_{k=1}^{|\mathcal{C}|} \exp\left(-Sim(c_k, c)\right)}. \tag{2}$$

γ_k is stored on the central server for fine-tuning subsequent global model updates.

3.2.3. Update local model

For a client participating in the federated training process, the client first receives the initialized/updated sequence encoder from the central server as its local model, then updates it based on local data, and finally sends the local weights to the server. The sequence encoder is our sequential recommendation model introduced in Section 3.3.

Given the client c_k and a set of users distributed on it, we adopt cross-entropy to define its average loss function:

$$\mathcal{L}(c_k) = \frac{1}{|\mathcal{U}_k|} \sum_{u \in \mathcal{U}_k} \sum_{v_i \in \mathcal{S}_u} \mathrm{loss}(u, v_i), \tag{3}$$

where $\mathrm{loss}(u, v_i)$ is the loss of observed interaction pair (u, v_i). For each observed interaction pair (u, v_i), we randomly sample K items $\{v_i^k\}_{k=1}^K$ that have not been interacted by user u, so the $\mathrm{loss}(u, v_i)$ can be defined as:

$$\mathrm{loss}(u, v_i) = -\log\left(\frac{\exp\left(\hat{y}(u, v_i)\right)}{\exp\left(\hat{y}\left(u, v_i\right)\right) + \sum_{k=1}^{K} \exp\left(\hat{y}\left(u, v_i^k\right)\right)}\right). \tag{4}$$

We adopt $\mathcal{L}(c_k)$ to derive the gradients of the local model and obtain its updated weights. Let w^k denote the parameter set of the local model in the kth client, w^k is updated by $\mathrm{w}_t^k = \mathrm{w}_{t-1}^k - \alpha \cdot \mathbf{g}_{t-1}^k$, where α and $\mathbf{g}_{t-1}^k$ are the learning rate and local gradients, respectively. Following [16], we apply the following local differential privacy (LDP) scheme with zero-mean Laplacian noise on w_t^k to enhance data privacy protection, formulated as:

$$\tilde{\mathrm{w}}_t^k = \mathrm{w}_t^k + \mathrm{Laplace}(0, \lambda), \tag{5}$$

where λ is Laplacian noise strength. The encrypted weights $\tilde{\mathrm{w}}_t^k$ are updated to the central server for further aggregation.

3.2.4. Update global model

The central server is responsible for the training process and the global model update. The central server first awakes some clients and sends them the initialized global sequence encoder. Then, the awakened clients update their local model parameters, encrypt and send them to the server. The central server aggregates the local models into a unified one and wakes up some clients again. This iterative process continues until the global model converges. Algorithm 1 summarizes the training process of FedSeqRec.

In FedAvg [17], the more examples of a client participating in training, the greater its "discursive power" in the global model update, which is formulated as:

$$w_{t+1} = \sum_{k \in S_t} \beta_k w_{t+1}^k \qquad \beta_k = \frac{n_k}{\sum_{k \in S_t} n_k}, \tag{6}$$

Algorithm 1 Training process of FedSeqRec

Input: $\mathcal{C}$, $\mathcal{U}_k$,c_k, **K**, **E**, δ,η
Initialize: $\mathbf{W} = \mathbf{W}_0$
1: The central server fits a polynomial distribution of $\mathbf{X}_s$ and gets $Q(s)$ based on test data
2: **for** $k = 1$ to $|\mathcal{C}|$, **Client do**
3: Learn $P(c_k)$ based on $\mathcal{U}_k$ locally and send it to the central server in parallel
4: **end for**
5: **for** $k = 1$ to $|\mathcal{C}|$, **Server do**
6: Compute Sim(c_k, s) by Eq. (1) and γ_k by Eq. (2)
7: **end for**
8: **for** $t = 1$ to T **do**
9: The central server awakes a subset of clients **K** from all clients randomly;
10: The central server sends $\mathbf{W}_{t-1}$ to all awaken clients;
11: Each client $k \in \mathbf{K}$ updates $\mathbf{W}_{t-1}^k$ for **E** epochs of SGD on $\mathcal{U}_k$ to obtain $\mathbf{W}_t^k$;
12: Each client $k \in \mathbf{K}$ sends β_k in Eq. (6) and noise-injected $\tilde{\mathbf{W}}_t^k$ to the central server in parallel;
13: The central server computes ζ_k for the kth client by Eq. (7);
14: The central server aggregates $\tilde{\mathbf{W}}_t^k$, $k \in \mathbf{K}$, and gets $\mathbf{W}_{t+1}$ by Eq. (7);
15: **end for**

where n_k is the number of training examples of client c_k, and S_t is the set of clients awakened in this iteration. However, if the local data distribution of clients deviates from the overall data distribution, the global model may be difficult to converge. Therefore, based on FedAvg, we further consider the impact of the difference between the client's local data distribution and the overall data distribution on the global model update. In our implementation, we propose a federated re-averaging algorithm formulated in Eq. (7) to aggregate local models.

$$\zeta_k = \delta\beta_k + (1-\delta)\,\gamma_k,\ \ 0 \le \delta \le 1$$
$$w_{t+1} = \sum_{k \in S_t} \zeta_k w_{t+1}^k \tag{7}$$

where β_k is presented in Eq. (6), γ_k is the adjustment factor in Eq. (2), δ is a hyperparameter, and ζ_k is the coefficient of the kth client parameter on the model update. The federated re-averaging algorithm degenerates to FedAvg if $\delta = 1$.

Eq. (7) reveals that if the clients' local data distribution is consistent with the overall distribution, different clients have the same ζ, which is equivalent to FedAvg [17]. If the client's local data distribution deviates from the overall distribution, the greater the deviation degree, the smaller the γ_k. In other words, γ_k reduces the client's "discourse power" in the global model update. To summarize, based on FedAvg, the federated re-averaging algorithm smoothes the impact of the global model updates when the client's local data distribution deviates from the overall data distribution.

3.3. Basic sequential recommendation model

This section describes the basic sequential recommendation model, *i.e.*,sequence encoder illustrated in Fig. 2. The sequence encoder aims to model the user's preferences.

3.3.1. Analysis on federated sequence modeling

The standard federated learning process adopts the "training locally, aggregating globally" cycle and proceeds it over time.

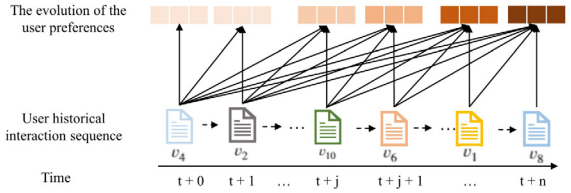


Fig. 3. Sequential user behavior modeling process. At the time $t+j$, we can learn the user preference by modeling his interaction sequence before time $t+j$, and the order of items in the interaction sequence is non-commutative.

When a sequential recommendation task encounters federated learning, modeling sequential user behavior along the time axis is natural. As depicted in Fig. 3, by modeling $[v_4, v_2]$ in which the position between v_4 and v_2 cannot be exchanged, we can obtain the user preferences at time $t+1$. Similarly, by modeling $[v_4, v_2, \ldots, v_{10}]$, in which the position of these items is non-commutative, we can obtain the user preferences at time $t+j$. It indicates that in federated sequential recommendation tasks, the non-commutativity of items' positions in the user–item interaction sequences plays a vital role. In conclusion, one salient feature of federated sequential recommendation is that the learning process is from left to right.

In mathematics, using non-commutative algebras to describe non-exchangeable sequences is a classic problem [21], and utilizing tensor projections to model the ordered structure affords promising performance [22]. Among existing representative sequential recommendation methods, RNN-based methods are insufficient to model long sequences due to the vanishing gradient problem [33]. The attention-based methods incorporating position information need to explicitly define position embedding functions [47] or inject learnable position embeddings [46]. Compared with RNN-based and attention-based methods, tensor projections methods can effectively deal with the long sequence representation and describe a sequence without explicitly defining or learning the position embeddings [22]. Therefore, the low-rank tensor projection is potentially suitable for sequential recommendation tasks.

3.3.2. The background of tensor algebras

Before introducing the sequence encoder in details, we first introduce the background of tensor algebras.

Tensor products. If $x \in \mathrm{R}^r$ and $y \in \mathrm{R}^m$ are two vectors, the tensor product between them, denoted as $x \otimes y$, is defined as the $(r \times m)$ -matrix which is a degree 2 tensor, with element $(x \otimes y)_{i,j} = x_i y_j$. Furthermore, if $z \in \mathrm{R}^n$ is another vector, their tensor product is a $(r \times m \times n)$ -matrix which is a degree 3 tensor, with elements $(x \otimes y \otimes z)_{i,j,k} = x_i y_j z_k$. Similarly, the tensor products can be extended to the form of infinite vector multiplication.

Tensor algebras. $\mathrm{T}(V)$ is the set of non-commutative sequences of tensors with all degrees:

$$\mathrm{T}(V) = \left\{ \mathbf{e} = (\mathbf{e}_n)_{n \ge 0} \mid \mathbf{e} \in V^{\otimes n} \right\},$$

where $V^{\otimes 0} = \mathrm{R}$, e_0 is a scalar, $\mathbf{e}_1$ and $\mathbf{e}_2$ are a vector and matrix, respectively, $\mathbf{e}_3$ is a 3-tensor, and so on.

Tensor linear space. For $\mathbf{u}, \mathbf{v} \in \mathrm{T}(V)$ and $\alpha \in \mathrm{R}$, the addition operation and scalar multiplication operation on $\mathrm{T}(V)$ are defined as:

$$\mathbf{u} + \mathbf{v} = (\mathbf{u}_n + \mathbf{v}_n)_{n \ge 0} \in \mathrm{T}(V) \text{ and } \alpha \cdot \mathbf{u} = (\alpha \mathbf{u}_n)_{n \ge 0} \in \mathrm{T}(V).$$

Since each $V^{\otimes n}$ is a linear space, a finite non-commutative sequence $(l_n)_{n=0}^N$ of tensors, $l_n \in V^{\otimes n}$, yields a linear functional on $\mathrm{T}(V)$ which can be expressed as:

$$\langle \mathbf{l}, \mathbf{v} \rangle = \sum_{n=0}^{N} \langle l_n, \mathbf{v}_n \rangle .$$

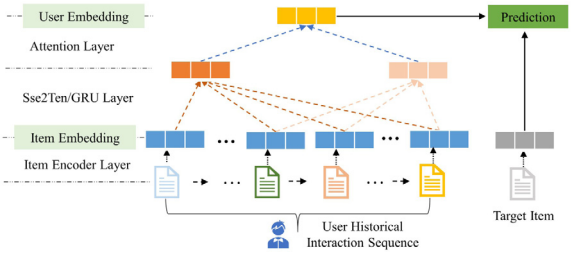


Fig. 4. The sequence encoder of FedSeqRec. For user u, the sequence encoder maps the interaction sequence to the user embedding e_u and returns a prediction score for a target item.

Non-commutative product. The non-commutative product on $\mathrm{T}(V)$ is the key to succeeding in modeling the sequence dependencies in [22]. For $\mathbf{p}, \mathbf{q} \in \mathrm{T}(V)$, the non-commutative product is defined as follows:

$$\begin{aligned}\mathbf{p}\cdot\mathbf{q} &= \left(\sum_{i=0}^{n}\mathbf{p}_i\otimes\mathbf{q}_{n-i}\right)_{n\geq 0}\\ &= (1, \mathbf{p}_1+\mathbf{q}_1, \mathbf{p}_2+\mathbf{p}_1\otimes\mathbf{q}_1+\mathbf{q}_2, \ldots),\end{aligned}$$

where $\otimes$ is the tensor product. Note that the product does not satisfy the commutative law, *i.e.*, $\mathbf{p}\cdot\mathbf{q} \neq \mathbf{q}\cdot\mathbf{p}$.

Non-commutative polynomials. A polynomial can be uniquely determined by a tensor algebra and vice versa. For instance, by replacing $B \mapsto (1+b)$, $C \mapsto (1+c)$, $D \mapsto (1+d)$, we have:

$$\begin{aligned}\text{“BBCD”} \mapsto &(1+b)(1+b)(1+c)(1+d)\\ &= 1+2b+c+d+bb+2bc+2bd\\ &+ cd+bbc+bbd+2bcd+bbcd.\end{aligned}$$

One may read off all the sub-sequence in “BBCD” from the coefficients of the polynomial on the right side of the above equation, *e.g.*,the term $2bc$ means that the sub-sequence “BC” appears twice. Given these definitions, we denote the item embeddings as the elements of the tensor space and model the user-dynamic preferences utilizing the non-commutative product.

3.3.3. Sequence encoder

As illustrated in Fig. 4, the proposed sequence encoder comprises the item encoder, seq2ten/GRU, attention, and prediction layers.

Item Encoder Layer. The item encoder consists of the input, self-attention, and attention layers.

Input Layer. The input layer aims to convert the attribution of an item into a series of semantic word embedding vectors. As depicted in Fig. 4, the original input of the model is the attributions of items that contain rich semantic information. For example, the movie's attributions include name, genre, director, and actors. For online learning videos, the knowledge concepts contained in the video constitute their attributions. Specifically, we use the 300-dimensional Global Vectors [48] for word representation to initialize word embedding, $\mathbf{E}_w = \mathrm{GloVe}(\mathcal{I})$ where $\mathcal{I}$ denotes the set of item attributions.

Self-Attention Layer. We adapt the self-attention mechanism [49] to learn the relationship between different attributions. As illustrated in Eq. (8), the following dot-product attention is used as the attention function:

$$\begin{aligned}&\mathrm{Score}(Q) = Q\odot\mathbf{E}_w\\ &\mathrm{Att}(Q,K) = \mathrm{softmax}\left(\frac{\mathrm{Score}(Q)\cdot\mathrm{Score}(K)^T}{\sqrt{d_k}}\right)\\ &\mathbf{E}_w^{new} = \mathrm{Att}(Q,K)\odot\mathrm{Score}(V),\end{aligned} \tag{8}$$

where $\mathbf{E}_w$ denotes the item attribution word embedding, $\odot$ is the element-wise product, Q, K, V are learnable parameters, $\sqrt{d_k}$ is the dimension of K, and $\mathbf{E}_w^{new}$ is the updated item attribution word embedding.

Attention Layer. An attention network is adopted to learn item embeddings e_v from the output of the above self-attention layer by extracting informative attributions.

Seq2Ten Layer. Given a user–item interaction sequence:

$$\mathcal{S}_u = \left[v_1^u, v_2^u, \ldots, v_{u_n}^u\right] \qquad \forall u\in\mathcal{U} \qquad \forall v_i^u \in R^d, \tag{9}$$

where $u_n \geq 1$ is the length of user u's clicking logs. The Seq2Ten layer aims to model users' dynamic preferences, involving three core steps: (I) defining the mapping from the item embedding v_i^u to the tensor space, (II) establishing the projection $\Phi : \mathcal{S}_u \to \mathrm{T}(R^d)$ that models the non-commutative interaction sequence as an entity of the tensor space $\mathrm{T}(R^d)$, and (III) adopting low-rank tensor functions to approximate Φ.

First, $\forall v_i^u \in R^d$, we define a projection as $g : v_i^u \to \mathrm{T}(R^d)$. Let

$$g\left(v_i^u\right) = \left(0, v_i^u, 0^{\otimes 2}, 0^{\otimes 3}, \ldots\right), \tag{10}$$

then $g\left(v_i^u\right) \in \mathrm{T}(R)$. Let $\mathbf{1} = \left(1, 0, 0^{\otimes 2}, 0^{\otimes 3}, \ldots\right)$, we have $\mathbf{1} \in \mathrm{T}(R)$. According to the characteristics of tensor linear space, we know $\mathbf{1} + g\left(v_i^u\right) \in \mathrm{T}(R^d)$.

Second, $\forall \mathcal{S}_u$ where $u \in \mathcal{U}$, define a projection $h : \mathcal{S}_u \to$ a non-commutative polynomial.

$$h(\mathcal{S}_u) = \prod_{i=1}^{u_n}\left(1+g\left(v_i^u\right)\right), \tag{11}$$

where $\prod$ is defined as the non-commutative product in Section 3.3.2. We obtain $h(\mathcal{S}_u) = (h_m(\mathcal{S}_u))_{m\geq 0}$, as illustrated in Eq. (12) by directly calculating Eq. (11).

$$h_m(\mathcal{S}_u) = \sum_{1\leq i_1<\cdots<i_m\leq u_n} g\left(v_{i_1}^u\right)\otimes\cdots\otimes g\left(v_{i_m}^u\right) \in R^{\otimes m}. \tag{12}$$

From the above process, we obtain the analytical form of the projection $\Phi = h(g)$, where Φ_m equals h_m.

Third, we speed up the calculation of Φ. Due to $v_i^u \in R^d$, the capacity to describe and store an element of $R^{\otimes m}$ is too computationally expensive. Fortunately, low-rank approximations aiming to address the complexity have been widely used in practice. To reduce computational overhead, a rank-1 tensor $\mathbf{l} = \mathbf{l}_1 \otimes \cdots \otimes \mathbf{l}_m \in R^{\otimes m}$ which is a learnable linear function is employed to compute the expression: Third, we speed up calculating Φ. Due to $v_i^u \in R^d$, the capacity to describe and store an element of $R^{\otimes m}$ is too computationally expensive. Nevertheless, low-rank approximations that address the complexity have been widely used in practice. To reduce the computational overhead, a rank-1 tensor $\mathbf{l} = \mathbf{l}_1\otimes\cdots\otimes\mathbf{l}_m \in R^{\otimes m}$, which is a learnable linear function, is employed to compute the expression:

$$\langle \mathbf{l}, \Phi(\mathcal{S}_u)\rangle = \sum_{1\leq i_1<\cdots<i_m\leq L}\prod_{k=1}^{n_u}\left\langle \mathbf{l}_k, g\left(v_{i_k}^u\right)\right\rangle, \tag{13}$$

without explicitly calculating all $h(\boldsymbol{\mathcal{S}}_{\mathbf{u}}) = (h_m(\boldsymbol{\mathcal{S}}_{\mathbf{u}}))_{m\geq 0}$.

After the Seq2Ten layer, we model the long-term preference based on the user–item interaction sequence, which can be formulated as $\mathbf{E}_{u_l} = \Phi(\mathcal{S}_u)$.

GRU Layer. Gated Recurrent Unit (GRU) is an RNN refined model that eases the vanishing gradient problem [32]. Thus, we adopt the GRU layer to model users' short-term preferences. GRU can be expressed as:

$$\mathbf{h_t} = (1-\mathbf{z_t})\,\mathbf{h_{t-1}} + \mathbf{z_t}\hat{\mathbf{h}}_\mathbf{t}, \tag{14}$$

Table 2
The statistics of the benchmark datasets.

Dataset	# Clients	# Users	# Items	# Training Sessions	# Test Sessions
MoocData	16	14,173	3,0430	21,492	5,298
MovieData	2	9,217	7,862	10,536	2,646

where the candidate activation $\mathbf{z_t}$ and the update gate $\hat{h}_t$ are expressed as:

$$\begin{aligned}\mathbf{z_t} &= \sigma\left(W_z\boldsymbol{\mathcal{S}}_{\mathbf{ut}} + U_z\mathbf{h_{t-1}}\right)\\ \hat{\mathbf{h}}_t &= \tanh\left(W\boldsymbol{\mathcal{S}}_{\mathbf{ut}} + U\left(\mathbf{r_t}\odot\mathbf{h_{t-1}}\right)\right)\\ \mathbf{r_t} &= \sigma\left(W_r\boldsymbol{\mathcal{S}}_{\mathbf{ut}} + U_r\mathbf{h_{t-1}}\right).\end{aligned} \tag{15}$$

By applying the GRU layer, we model the users' short-term preferences, denoted as $\mathbf{E}_{u_s} = \mathrm{GRU}\left(\mathcal{S}_u\right)$.

Attention Layer. The attention layer is employed to aggregate the user's long-term preferences $\mathbf{E}_{u_l}$ and short-term preferences $\mathbf{E}_{u_s}$, as presented below:

$$\begin{aligned}\alpha_{ul} &= \frac{\exp\left(W_2^l\tanh\left(W_1^l\mathbf{E}_{u_l}+b_1^l\right)+b_2^l\right)}{\sum_{k\in\{l,s\}}\exp\left(W_2^k\tanh\left(W_1^k\mathbf{E}_{u_k}+b_1^k\right)+b_2^k\right)}\\ \alpha_{us} &= \frac{\exp\left(W_2^s\tanh\left(W_1^s\mathbf{E}_{u_s}+b_1^s\right)+b_2^s\right)}{\sum_{k\in\{l,s\}}\exp\left(W_2^k\tanh\left(W_1^k\mathbf{E}_{u_k}+b_1^k\right)+b_2^k\right)}\\ E_u &= \alpha_{ul}\mathbf{E}_{u_l}+\alpha_{us}\mathbf{E}_{u_s},\end{aligned} \tag{16}$$

where $\|$ denotes the concatenation operator, α_{ul} and α_{us} are weights of long-term preference and short-term preference, respectively. E_u is the user's overall preference embedding.

Prediction Layer. After obtaining the item embedding E_v and user embedding E_u, we can predict the score of any user–item pair by Eq. (17), which is the dot product of their embedding vectors.

$$\hat{y}_{ij} = \left(E_u[i]\right)^T\left(E_v[j]\right)\ \ \forall i \le |\mathcal{U}|\ \forall j \le |\mathcal{V}|, \tag{17}$$

where $E_u[i]$ is the embedding of the ith user, $E_v[j]$ is the embedding of the jth item.

3.4. Analysis on privacy protection

Data privacy can be protected from four aspects in FedSeqRec. First, the private user behavior data remains with the clients during the global model training process. Compared with centralized storage involving user interaction data, this architecture avoids privacy leakage during the data transmission. Second, before the model training begins, the operation clients deliver their local data distribution to the central server without exposing the user's raw behavior data. Even if the server obtains the length of a user's behavior sequence by chance, it still does not affect the raw interaction data. Third, during the model training, the local model parameters computed by the awakened clients communicate with the central server without disclosing the user privacy. This is because: (I) The local model parameters do not carry the user's raw private data. (II) Even if the server wakes up a client twice in a row and infers its gradients, the gradients contain much less sensitive information based on data processing inequality [17]. (III) The local model parameters are derived from a batch of users, making it difficult to infer the privacy behavior of a specific user. Fourth, we add Laplace noise to the local model parameters before uploading them to the server, enhancing the privacy protections of the behavior information in the clients. Since a slightly larger λ in Eq. (5) reduces the global model's accuracy, we adopt a very small λ to balance the global model performance and data privacy protection.

4. Experiments

This section assesses the effectiveness of the proposed method and answer the following questions about FedSeqRec: (**Q1**) How is its recommendation performance? (**Q2**) What is the contribution of each component? (**Q3**) What is the influence of the number of clients in the training process? (**Q4**) What is the impact of the Laplacian noise strength λ in Eq. (5)? (**Q5**) How is the convergence? Before answering these questions, we first introduce the experimental datasets, baseline methods, and evaluation metrics.

4.1. Experimental settings

4.1.1. Datasets

We collect three real-world datasets, *i.e.*, MOOCCube,[1] MovieLens[2] and MovieTweetings,[3] and pre-process them into two synthetic datasets to verify our developed FedSeqRec, *i.e.*, MoocData, and MovieData.

MoocData Pre-Processing. The MoocData contains processed user-video, course-video, school-course, and video-concept relations from the MOOCCube dataset. For video feature pre-processing, we choose the knowledge concepts contained in the video as attributes. Concerning user pre-processing, we consider that a user comes from a school where the videos are viewed most by the user. Hence, users from the kth school constitute the local dataset of the kth client, *i.e.*, $\mathcal{U}_k$. For label pre-processing, MOOCCube records the duration of the user watching videos, ranging from a few seconds to dozens of minutes. This work removes the user-video watching pairs less than 5 min long. Therefore, we construct implicit feedback as 1 for the user-video pairs where a user watches a video for more than 5 min and 0 for the rest.

MovieData Pre-Processing. The MovieData comes from the processed MovieLens and MovieTweetings datasets. For movie feature pre-processing, we consider the movie attributes including the genre, title, and year the movie was released. In addition, we find the movie intersection contained in MovieLens and MovieTweetings, then select the respective user interaction sequences in both datasets based on these movies, which naturally constitute the local datasets of the MovieLens client and the MovieTweetings client. For label pre-processing, the user's rating range for the movie in MovieLens is 1 to 5 and 1 to 10 in MovieTweetings. A higher rating indicates that the user is more likely to prefer the movie. In this work, we construct implicit feedback as 1 for those ratings not less than 3 in MovieLens or not less than 7 in MovieTweetings and 0 for the other ratings.

We treat the sequence within a month as a session and discard the sessions having only one interaction. Table 2 presents the detailed statistics of the processed datasets.

4.1.2. Baseline methods

We challenge the proposed FedSeqRec against the following recommendation methods:

[1] http://moocdata.cn/data/MOOCCube
[2] https://grouplens.org/datasets/movielens/
[3] https://github.com/sidooms/MovieTweetings

- **GRU4Rec** [32]: This RNN-based approach uses GRU with a ranking loss to model user behavior sequences.
- **Bert4Rec** [44]: It is an attention-based approach that uses the deep bidirectional self-attention to model user interaction sequences.
- **SR-GNN** [38]: A GNN-based approach that represents a session as the combination of the global preference and the current interest of that session.
- **FCF** [12]: This is the first federated collaborative filtering method for a recommendation based on users' implicit feedback.
- **FedNewsRec** [16]: An attention-based federated recommendation method customized for news recommendation.
- **FedRec++** [14]: This is a PMF-based [28] lossless federated recommendation method by assigning different roles for clients.

4.1.3. Evaluation metrics

To evaluate FedSeqRec, we employ the leave-one-out evaluation method [50–52]. Specifically, for each session, we keep the last item of the session as the test data, and others are used for training. For each test item, we randomly sample M negative samples. Hence, the sequential recommendation task becomes a ranking problem that ranks these negative samples with the target item.

We utilize a variety of popular evaluation metrics to measure the quality of the ranking list, including Area Under Curve (AUC), Mean Reciprocal Rank (MRR), and Normalized Discounted Cumulative Gain (NDCG). The definitions of AUC, MRR, NDCG@K are shown in Eq. (18), Eq. (19), Eq. (20), respectively.

$$AUC = \frac{\sum_{\text{positives}} \text{rank} - \frac{H\times(H+1)}{2}}{H * M}, \tag{18}$$

where H and M are negative and positive samples, respectively. $\sum_{\text{positives}}$ is the sum of position numbers of positive samples after the prediction scores are in ascending order.

$$MRR = \frac{1}{|\mathcal{U}|}\sum_{i=1}^{|\mathcal{U}|}\frac{1}{\text{rank}_i} \tag{19}$$

where $|\mathcal{U}|$ is the number of users, rank_i is the position of the target item in the recommended list to the ith user.

$$\begin{aligned} NDCG@K &= \frac{1}{U}\sum_{u=1}^{U}\frac{DCG_u@K}{IDCG_u@K}, \\ DCG_u@K &= \sum_{i=1}^{K}\frac{2^{rel_u^i}-1}{\log_2(i+1)}, \end{aligned} \tag{20}$$

where $IDCG_u@K$ is the ideal discounted cumulative gain for the uth user recommendation list, rel_u^i is the item graded relevance ranked at position i to the uth user.

4.1.4. Parameter settings

The dimensions of the user and item embeddings are set to 300, the same as the dimension of pre-trained Glove word embeddings. The learning rate of MoocData and MovieData is 0.40 and 0.35, respectively. The fraction of clients participating in model training in each round is 30% for MoocData, while the two clients on MovieData, *i.e.*, MovieLens, and MovieTweetings, participate in model training in every round. As for hyperparameter δ in Eq. (7), we empirically set it to 0.2 in MoocData and 0.3 in MovieData. For GRU4Rec, Bert4Rec, SR-GNN, FedNewsRec, and FedRec++, we employ the code provided by the corresponding authors, while FCF is implemented using Python. We utilize the embedding dimension present in the original models and focus on tuning the learning rate. Finally, we record the average results of five experiments under the same experimental setup.

4.2. Overall performance comparison (**RQ1**)

We compare the recommendation accuracy of FedSeqRec against the centralized and the federated recommendation models. Similar to [12], we also compute "Improv". (see Eq. (21)) between the performance metrics of the different models.

$$\text{Improv.} = \frac{\text{Metric}(\text{Model}_1) - \text{Metric}(\text{Model}_2)}{\text{Metric}(\text{Model}_2)}. \tag{21}$$

The results are reported in Table 3, highlighting that:

- The proposed FedSeqRec exhibits a competitive recommendation performance. Specifically, none of the competitor methods simultaneously achieve state-of-the-art performance on MoocData and MovieData. Considering the non-federated baselines, FedSeqRec improves the best performance by 7.98% in MRR and 6.14% in NDCG@5 on MovieData. For the federated scenes, FedSeqRec boosts the best one by 4.45% and 1.18% in terms of NDCG@5 on MoocData and MovieData, respectively.
- Regarding the centralized methods, Bert4Rec achieves the best recommendation performance, followed by SR-GNN, and GRU4Rec presents the least appealing performance. FedSeqRec is superior to SR-GNN and GRU4Rec on both datasets and is better than Bert4Rec on MovieData. Although Bert4Rec performs slightly better than FedSeqRec on MoocData, FedSeqRec is still competitive because the performance of a model using federated training will approximate that of the model using centralized training. In practice, federated training may incur some accuracy loss, which is a trade-off between accuracy and privacy. More importantly, Bert4Rec is not compatible with the real FL scenario. As discussed in Section 3.3.1, learning user preferences from right to left inverse time in the FL framework is unrealistic, which is the key to Bert4Rec's success.
- In decentralized methods, FedNewsRec is better than FCF and FedRec++ because FedNewsRec adopts an attention mechanism to learn the item dependence in sequences. Note that FedSeqRec achieves higher recommendation accuracy than FedNewsRec on the two datasets, verifying the effectiveness of the Seq2Ten layer in modeling the users' long-term preferences in sequential recommendation tasks.
- For a fair comparison, we perform centralized training on our developed algorithm and the best-federated baseline. The experimental results are summarized in Table 4, demonstrating that when adopting the centralized training method, FedSeqRec is superior to FesNewsRec on both benchmark datasets. Additionally, FedSeqRec is better than Bert4Rec on MoocDate and achieves comparable results with Bert4Rec on MoocData, verifying our method's performance. Finally, FedSeqRec has 25% and 48% less parameters than FesNewsRec and Bert4Rec respectively, making FedSeqRec more attractive for the participants in FL.

4.3. Ablation study (**RQ2**)

This section performs ablation experiments on FedSeqRec to investigate the contributions of the federated re-average algorithm and the low-rank tensor projections. The recommendation performance achieved by different FedSeqRec variants is presented in Fig. 5, from which we make the following observations.

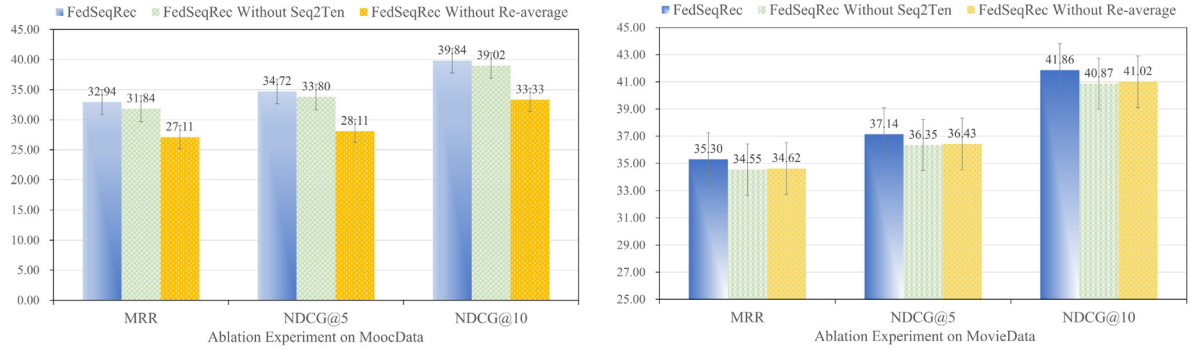
- When disabling the federated re-average technique, δ in Eq. (7) is 1. It is equivalent to the FedAvg [17]. The federated re-averaging technique boosts the NDCG@5 by about

Table 3
The recommendation performance comparison of different methods on two benchmark datasets. The best and the second-best results are marked ** and *, respectively. *Improv.* is the percentage difference between the mean values of FedSeqRec and FesNewsRec that achieves the best results in the federated recommendation.

		MoocData				MovieData			
Methods	Federated-based	AUC	MRR	NDCG@5	NDCG@10	AUC	MRR	NDCG@5	NDCG@10
GRU4Rec	No	71.54	27.44	29.13	33.44	68.74	28.12	30.03	34.48
Bert4Rec	No	97.35**	35.21**	37.16**	42.62**	73.47	32.69	34.99	38.23
SR-GNN	No	83.82	28.94	31.26	35.98	72.93	31.09	32.46	36.75
FCF	Yes	70.04	23.68	25.02	31.72	66.94	27.10	28.47	33.66
FedNewsRec	Yes	96.84	31.75	33.24	38.47	85.26*	34.89*	36.71*	41.09*
FedRec++	Yes	69.93	22.51	23.34	30.99	65.20	25.14	26.47	31.49
FedSeqRec	Yes	96.92*	32.94*	34.72*	39.84*	86.32**	35.30**	37.14**	41.86**
Improv.(%)		0.08	3.76	4.45	3.56	1.24	1.19	1.18	1.89

Table 4
The model parameters and accuracy when training the sequence encoder in a centralized way.

		MoocData				MovieData			
Methods	#Parameters	AUC	MRR	NDCG@5	NDCG@10	AUC	MRR	NDCG@5	NDCG@10
Bert4Rec	3,621,248	97.35	35.21	37.16	42.62	73.47	32.69	34.99	38.23
FesNewsRec	2,524,003	97.08	33.05	34.80	39.97	85.97	35.41	37.42	41.89
FedSeqRec	1,884,406	97.17	34.62	36.64	41.76	86.88	36.19	38.04	42.54



(a) MoocData (b) MovieData

Fig. 5. The recommendation performance of different variants of FedSeqRec on MoocData and MovieData datasets.

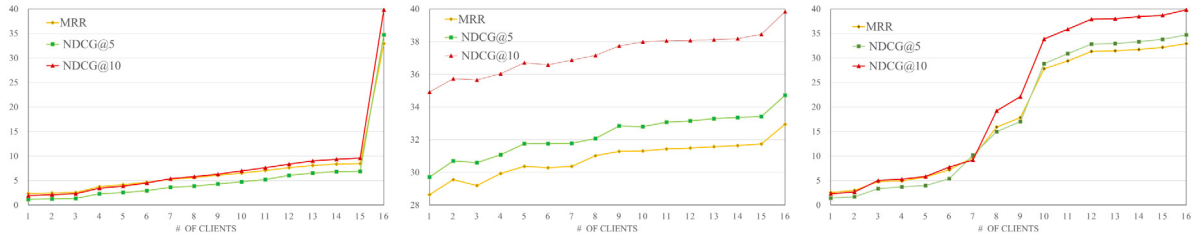
23.5% and 1.9% on MoocData and MovieData, respectively. The local data distribution of MoocData is more divergent than that of MovieData. Correspondingly, the contribution of the federated re-average scheme in MoocData is about 12.4 times of that in MovieData, indicating that the federated re-average algorithm smoothes the impact of the global model updates when the client's local data distribution deviates from the overall data distribution.

- When disabling the low-rank tensor projection, α_{ul} in Eq. (16) is 0. We find that the low-rank tensor projection gains NDCG@5 by about 1.9% and 2.2% on the MoocData and MovieData, respectively. The experimental results indicate that FedSeqRec benefits from the low-rank tensor projection on the two datasets, highlighting that the low-rank tensor projection is an alternative method to model dynamic user preferences in recommender systems. These findings validate our motivations introduced in Section 3.3.1.

4.4. *Influence of client number* (**RQ3**)

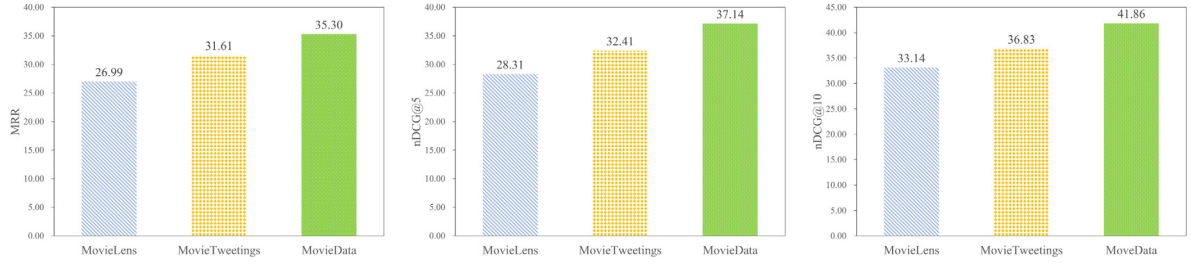
To explore whether FedSeqRec can utilize the user interaction information in different clients federated to train an effective sequential recommendation model, we arrange different numbers of clients to participate in training and employ all clients for evaluation. For MoocData, we sort the clients based on their amount of data in ascending order and number them from 1 to 16. We conduct three experiments to explore the impact of the number of clients taking part in training: (I) Adding clients in ascending order of client number. This means that the first k clients perform the model training involving k clients. (II) Adding clients in descending order of client number. This means the last k clients complete the model training involving k clients. (III) Adding clients by random sampling, meaning that the model training involving k clients is completed by randomly sampling k clients without replacement from all clients. The experimental results of MoocData are illustrated in Fig. 6. For MovieData, we employ the local data of MovieLens and MovieTweetings to train the global model. The results are depicted in Fig. 7.

- The three subgraphs in Fig. 6 highlight the same trend. Looking along the x-axis in the subgraphs, it is evident that when the number of clients participating in training is small, the performance of FedSeqRec is unsatisfactory. The potential reason is that a client with relatively few users is not enough to train a deep model. As the number of clients joining in model training increases, the accuracy of our model improves. This observation indicates that our model can effectively learn knowledge from user interaction sequences from different clients to jointly train a competitive federated sequence recommendation model, verifying FedSeqRec's effectiveness. In addition, the curve's inflection



(a) Training in ascending order of client number (b) Training in descending order of client number (c) Training by sampling clients randomly

Fig. 6. The performance of FedSeqRec with different numbers of clients on MoocData in terms of MRR, NDCG@5, and NDCG@10.



(a) Performance on MRR (b) Performance on NDCG@5 (c) Performance on NDCG@10

Fig. 7. The performance of FedSeqRec concerning the local data on only one client, *i.e.*,MovieLens or MovieTweetings, in terms of MRR, NDCG@5, and NDCG@10.

point position in Fig. 6 corresponds to the client with the most local data, indicating that different clients have different contributions to training the global model. The clients' roles in global model training will be investigated in future work.

- As depicted in Fig. 7, in the framework of federated sequential recommendation involving two clients, the performance of the global model trained using the local data of one client is always weaker than the performance of collective training. This highlights that FedSeqRec has learned additional "knowledge" that does not belong to a single client, indicating that the FedSeqRec proposed is practical.

4.5. Hyper-parameter analysis (**RQ4**)

We conduct experiments on MoocData and MovieData to explore the influence of the hyper-parameter λ on the proposed method. Specifically, λ in Eq. (5) controls the Laplacian noise strength of the local differential privacy in the proposed FedSeqRec framework.

The results are illustrated in Fig. 8, revealing that the FedSeqRec's performance on the two benchmark datasets declines as the λ value increases. Larger λ means stronger Laplace noise added to local weights, making the weights for global model aggregation less accurate. Note that the larger λ, the stronger the privacy protection of the FedSeqRec. In other words, better privacy protection is at the expense of accuracy. Therefore, we set an appropriate λ value to trade off privacy protection and recommendation performance.

4.6. Convergence analysis (**RQ5**)

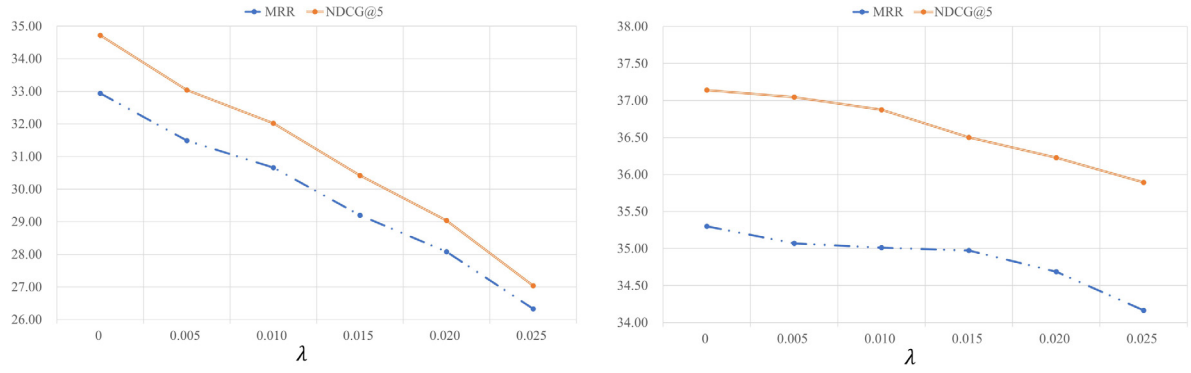
We explore the convergence of FedSeqRec and report the results. The AUC and the global training loss of FedSeqRec for different δ introduced in Eq. (7) on MoocData and MovieData are depicted in Fig. 9 and Fig. 10, respectively.

- From Fig. 9, we have three observations. First, from a holistic perspective, the training process can converge in about 200 rounds under different settings of δ for MoocData and MovieData, respectively. This indicates that FedSeqRec can train the sequential recommendation model efficiently. Secondly, from a comparative perspective, the convergence curve shows differences with various δ. It implies that a gentle δ helps speed up the model convergence, thus verifying the effectiveness of the federated re-averaging algorithm introduced in Section 3.2. Third, FedSeqRec is simultaneously effective on MoocData and MovieData, showing its generalization ability.
- As depicted in Fig. 10, the global training loss converges under different learning rates on MoocData and MovieData, further proving the effectiveness of the proposed federated sequential recommendation method.

5. Conclusion and future work

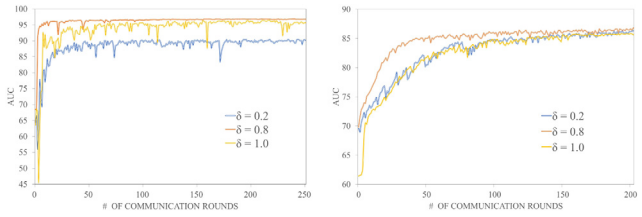
This work proposes a novel horizontal federated framework for sequential recommendation named FedSeqRec. The participants in FedSeqRec are organizations, and the developed scheme differs from the existing methods that rely on either centralized or distributed model training directly to mobile users. Data in different organizations tend to be Non-IID, imposing the global model to converge to sub-optimal results. To alleviate the Non-IID problem, we propose a federated re-averaging algorithm for model training and employ a low-rank tensor projection method to model the user-dynamic preferences. To the best of our knowledge, this is the first attempt to utilize an algebra-based approach to model user preferences in recommender systems. Experiments on two preprocessed real-world datasets show that FedSeqRec achieves competitive recommendation accuracy with fewer trainable parameters than several centralized and decentralized baselines.

For future work, we intend to advance FedSeqRec in the following directions. A valuable point is relaxing the constraints on the existing framework, allowing the users of different clients to be shared, *e.g.*, user *Bob* can register MovieLens and MovieTweetings simultaneously. When considering more items, learning all the item embeddings is expensive. Thus, another potential research is adopting the pre-training technology to learn item



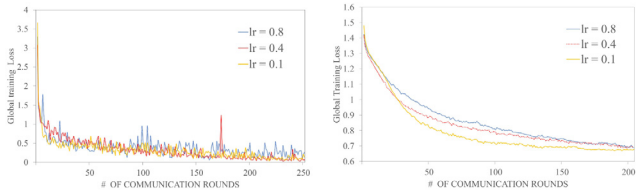
(a) MoocData (b) MovieData

Fig. 8. The influence of the hyper-parameter λ, *i.e.*, Laplacian noise, on the performance of FedSeqRec.



(a) MoocData (b) MovieData

Fig. 9. The convergence of FedSeqRec training for MoocData and MovieData in terms of AUC.



(a) MoocData (b) MovieData

Fig. 10. The trends of global training loss of FedSeqRec with different learning rates.

embeddings and feed them to subsequent user encoder modules [53], thus reducing the deployment and communication costs of the global model for participants in FL. In addition, we will also consider other methods to measure the similarity, such as the latent variables that encode user sequences or the out-degree of user nodes in a user–item interaction graph.

CRediT authorship contribution statement

Li Li: Methodology, Data curation, Software, Writing – original draft. **Fan Lin:** Resources, Validation. **Jianbing Xiahou:** Supervision, Funding acquisition. **Yuanguo Lin:** Visualization. **Pengcheng Wu:** Investigation, Project administration. **Yong Liu:** Conceptualization, Writing – review & editing, Formal analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This research is supported by the National Natural Science Foundation of China (No. 61977055).

References

[1] W. Dianshuang, L. Jie, Z. Guangquan, A fuzzy tree matching-based personalized E-learning recommender system, IEEE Trans. Fuzzy Syst. 23 (6) (2015) 2412–2426.
[2] G. Jibing, W. Shen, W. Jinlong, F. Wenzheng, P. Hao, T. Jie, S. Yu Philip, Attentional graph convolutional networks for knowledge concept recommendation in MOOCs in a heterogeneous view, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 79–88.
[3] Y. Lin, S. Feng, F. Lin, W. Zeng, Y. Liu, P. Wu, Adaptive course recommendation in MOOCs, Knowl.-Based Syst. 224 (2021) 107085.
[4] Y. Liu, W. Wei, A. Sun, C. Miao, Exploiting geographical neighborhood characteristics for location recommendation, in: Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management, 2014, pp. 739–748.
[5] P. Han, Z. Li, Y. Liu, P. Zhao, J. Li, H. Wang, S. Shang, Contextualized point-of-interest recommendation, in: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, 2020.
[6] C. Lei, Y. Liu, L. Zhang, G. Wang, H. Tang, H. Li, C. Miao, SEMI: A sequential multi-modal information transfer network for E-commerce micro-video recommendations, in: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2021.
[7] S. Zhang, L. Yao, A. Sun, Y. Tay, Deep learning based recommender system: A survey and new perspectives, ACM Comput. Surv. 52 (1) (2019) 1–38.
[8] P. Voigt, A. Von dem Bussche, The eu general data protection regulation (gdpr), in: A Practical Guide, first ed., Vol. 10, (1) Springer International Publishing, Cham, 2017, 3152676.
[9] Y. Qiang, L. Yang, C. Tianjian, T. Yongxin, Federated machine learning: Concept and applications, ACM Trans. Intell. Syst. Technol. (TIST) 10 (2) (2019) 1–19.
[10] Z. Chen, X. Yu, B. Hang, Y. Bin, L. Weihong, G. Yuan, A survey on federated learning, Knowl.-Based Syst. 216 (2021) 106775.
[11] L. Tian, S.A. Kumar, T. Ameet, S. Virginia, Federated learning: Challenges, methods, and future directions, IEEE Signal Process. Mag. 37 (3) (2020) 50–60.
[12] A.-U.-D. Muhammad, I. Elena, S.A. Khan, O. Were, F. Qiang, T.K. Eeik, F. Adrian, Federated collaborative filtering for privacy-preserving personalized recommendation system, 2019, arXiv preprint arXiv:1901.09888.
[13] C. Di, W. Leye, C. Kai, Y. Qiang, Secure federated matrix factorization, IEEE Intell. Syst. (2020).
[14] L. Feng, P. Weike, M. Zhong, FedRec++: Lossless federated recommendation with explicit feedback, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, 5, 2021, pp. 4224–4231.
[15] Y. Enyue, H. Yunfeng, L. Feng, P. Weike, M. Zhong, FCMF: Federated collective matrix factorization for heterogeneous collaborative filtering, Knowl.-Based Syst. 220 (2021) 106946.
[16] Q. Tao, W. Fangzhao, W. Chuhan, H. Yongfeng, X. Xing, Privacy-preserving news recommendation model learning, in: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings, 2020, pp. 1423–1432.
[17] M. Brendan, M. Eider, R. Daniel, H. Seth, y Arcas Blaise Aguera, Communication-efficient learning of deep networks from decentralized data, in: Artificial Intelligence and Statistics, 2017, pp. 1273–1282.
[18] P. Liangming, L. Chengjiang, L. Juanzi, T. Jie, Prerequisite relation learning for concepts in moocs, in: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2017, pp. 1447–1456.

[19] J. Zhang, C. Gao, D. Jin, Y. Li, Group-buying recommendation for social e-commerce, in: 2021 IEEE 37th International Conference on Data Engineering (ICDE), IEEE, 2021, pp. 1536–1547.
[20] G. Zhou, N. Mou, Y. Fan, Q. Pi, W. Bian, C. Zhou, X. Zhu, K. Gai, Deep interest evolution network for click-through rate prediction, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 01, 2019, pp. 5941–5948.
[21] C. Reutenauer, Free lie algebras, in: Handbook of Algebra, Vol. 3, Elsevier, 2003, pp. 887–903.
[22] C. Toth, P. Bonnier, H. Oberhauser, Seq2Tens: An efficient representation of sequences by low-rank tensor projections, in: International Conference on Learning Representations, 2020.
[23] P. Kairouz, H.B. McMahan, B. Avent, A. Bellet, M. Bennis, A.N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al., Advances and open problems in federated learning, Found. Trends Mach. Learn. 14 (1–2) (2021) 1–210.
[24] G. Lin, F. Liang, W. Pan, Z. Ming, Fedrec: Federated recommendation with explicit feedback, IEEE Intell. Syst. (2020).
[25] H. István, D. Gábor, J. Márk, Decentralized recommendation based on matrix factorization: a comparison of gossip and federated learning, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, 2019, pp. 317–332.
[26] V.W. Anelli, Y. Deldjoo, T.D. Noia, A. Ferrara, F. Narducci, Federank: User controlled feedback with federated recommender systems, in: European Conference on Information Retrieval, 2021, pp. 32–47.
[27] J.L. Herlocker, J.A. Konstan, L.G. Terveen, J.T. Riedl, Evaluating collaborative filtering recommender systems, ACM Trans. Inf. Syst. (TOIS) 22 (1) (2004) 5–53.
[28] M. Andriy, R. Salakhutdinov Russ, Probabilistic matrix factorization, Adv. Neural Inf. Process. Syst. 20 (2007) 1257–1264.
[29] F. Hui, Z. Danning, S. Yiheng, G. Guibing, Deep learning for sequential recommendation: Algorithms, influential factors, and evaluations, ACM Trans. Inf. Syst. (TOIS) 39 (1) (2020) 1–42.
[30] M. Quadrana, P. Cremonesi, D. Jannach, Sequence-aware recommender systems, ACM Comput. Surv. 51 (4) (2018) 1–36.
[31] S. Wang, L. Cao, Y. Wang, Q.Z. Sheng, M.A. Orgun, D. Lian, A survey on session-based recommender systems, ACM Comput. Surv. 54 (7) (2021) 1–38.
[32] H. Balázs, K. Alexandros, B. Linas, T. Domonkos, Session-based recommendations with recurrent neural networks, in: International Conference on Learning Representations, 2016.
[33] H. Sepp, S. Jürgen, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[34] Q. Massimo, K. Alexandros, H. Balázs, C. Paolo, Personalizing session-based recommendations with hierarchical recurrent neural networks, in: Quadrana, Massimo and Karatzoglou, Alexandros and Hidasi, Balázs and Cremonesi, Paolo, 2017, pp. 130–137.
[35] W. Chao-Yuan, A. Amr, B. Alex, J. Smola Alexander, J. How, Recurrent recommender networks, in: Proceedings of the Tenth ACM International Conference on Web Search and Data Mining, 2017, pp. 495–503.
[36] Y. Feng, L. Qiang, W. Shu, W. Liang, T. Tieniu, A dynamic recurrent model for next basket recommendation, in: Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2016, pp. 729–732.
[37] D. Tim, L. Benedikt, Z. Jürgen, Sequential user-based recurrent neural network recommendations, in: Proceedings of the Eleventh ACM Conference on Recommender Systems, 2017, pp. 152–160.
[38] W. Shu, T. Yuyuan, Z. Yanqiao, W. Liang, X. Xing, T. Tieniu, Session-based recommendation with graph neural networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 01, 2019, pp. 346–353.
[39] X. Chengfeng, Z. Pengpeng, L. Yanchi, S. Sheng Victor, X. Jiajie, Z. Fuzhen, F. Junhua, Z. Xiaofang, Graph contextualized self-attention network for session-based recommendation, in: IJCAI, Vol. 19, 2019, pp. 3940–3946.
[40] P. Zhiqiang, C. Fei, L. Yanxiang, d.R. Maarten, Rethinking item importance in session-based recommendation, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 1837–1840.
[41] L. Jing, R. Pengjie, C. Zhumin, R. Zhaochun, L. Tao, M. Jun, Neural attentive session-based recommendation, in: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, 2017, pp. 1419–1428.
[42] L. Qiao, Z. Yifu, M. Refuoe, Z. Haibin, STAMP: Short-term attention/memory priority model for session-based recommendation, in: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2018, pp. 1831–1839.
[43] K. Wang-Cheng, M. Julian, Self-attentive sequential recommendation, in: 2018 IEEE International Conference on Data Mining (ICDM), 2018, pp. 197–206.
[44] S. Fei, L. Jun, W. Jian, P. Changhua, L. Xiao, O. Wenwu, J. Peng, BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer, in: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, 2019, pp. 1441–1450.
[45] Y. Feng, Z. Yanqiao, L. Qiang, W. Shu, W. Liang, T. Tieniu, TAGNN: Target attentive graph neural networks for session-based recommendation, in: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 1921–1924.
[46] K. Wang-Cheng, M. Julian, Self-attentive sequential recommendation, in: 2018 IEEE International Conference on Data Mining (ICDM), 2018, pp. 197–206.
[47] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in Neural Information Processing Systems, 2017, pp. 5998–6008.
[48] P. Jeffrey, S. Richard, D. Manning Christopher, Glove: Global vectors for word representation, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014, pp. 1532–1543.
[49] V. Ashish, S. Noam, P. Niki, U. Jakob, J. Llion, N. Gomez Aidan, K. Lukasz, P. Illia, Attention is all you need, in: Advances in Neural Information Processing Systems 30 (NIPS 2017), 2017, pp. 5998–6008.
[50] B. Immanuel, H. Xiangnan, K. Bhargav, R. Steffen, A generic coordinate descent framework for learning from implicit feedback, in: Proceedings of the 26th International Conference on World Wide Web, 2017, pp. 1341–1350.
[51] T. Jiaxi, W. Ke, Personalized top-n sequential recommendation via convolutional sequence embedding, in: Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, 2018, pp. 565–573.
[52] H. Xiangnan, L. Lizi, Z. Hanwang, N. Liqiang, H. Xia, C. Tat-Seng, Neural collaborative filtering, in: Proceedings of the 26th International Conference on World Wide Web, 2017, pp. 173–182.
[53] V.W. Anelli, T. Di Noia, E. Di Sciascio, A. Ferrara, A.C.M. Mancino, Sparse feature factorization for recommender systems with knowledge graphs, in: Fifteenth ACM Conference on Recommender Systems, 2021, pp. 154–165.